Activity Report 2011

Section Software

Edition: 2012-03-23
<table>
<thead>
<tr>
<th>Algorithmics, Programming, Software and Architecture</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. LFANT Project-Team</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Applied Mathematics, Computation and Simulation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2. ALEA Project-Team (section vide)</td>
<td>6</td>
</tr>
<tr>
<td>3. BACCHUS Team</td>
<td>7</td>
</tr>
<tr>
<td>4. CAGIRE Team</td>
<td>12</td>
</tr>
<tr>
<td>5. CONCHA Project-Team</td>
<td>13</td>
</tr>
<tr>
<td>6. CQFD Project-Team</td>
<td>18</td>
</tr>
<tr>
<td>7. GEOSTAT Project-Team</td>
<td>19</td>
</tr>
<tr>
<td>8. MC2 Project-Team</td>
<td>20</td>
</tr>
<tr>
<td>9. REALOPT Project-Team</td>
<td>22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Computational Sciences for Biology, Medicine and the Environment</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10. ANUBIS Project-Team (section vide)</td>
<td>23</td>
</tr>
<tr>
<td>11. CARMEN Team (section vide)</td>
<td>24</td>
</tr>
<tr>
<td>12. MAGIQUE-3D Project-Team</td>
<td>25</td>
</tr>
<tr>
<td>13. MAGNOME Project-Team</td>
<td>27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Networks, Systems and Services, Distributed Computing</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>14. CEPAGE Project-Team</td>
<td>29</td>
</tr>
<tr>
<td>15. HIEPACS Project-Team</td>
<td>31</td>
</tr>
<tr>
<td>16. PHOENIX Project-Team</td>
<td>35</td>
</tr>
<tr>
<td>17. RUNTIME Project-Team</td>
<td>40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Perception, Cognition, Interaction</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>18. FLOWERS Project-Team</td>
<td>45</td>
</tr>
<tr>
<td>19. IPARLA Project-Team</td>
<td>62</td>
</tr>
</tbody>
</table>
5. Software

5.1. Pari/Gp

Participants: Karim Belabas [correspondant], Bill Allombert, Henri Cohen, Andreas Enge.

http://pari.math.u-bordeaux.fr/

PARI/GP is a widely used computer algebra system designed for fast computations in number theory (factorisation, algebraic number theory, elliptic curves, ...), but it also contains a large number of other useful functions to compute with mathematical entities such as matrices, polynomials, power series, algebraic numbers, etc., and many transcendental functions.

- PARI is a C library, allowing fast computations.
- GP is an easy-to-use interactive shell giving access to the PARI functions.
- gp2c, the GP-to-C compiler, combines the best of both worlds by compiling GP scripts to the C language and transparently loading the resulting functions into GP; scripts compiled by gp2c will typically run three to four times faster.

2011 has seen the release of the next major stable version, 2.5, ending the 2.3 release series started in 2007.

- Version of PARI/GP: 2.5.0
- Version of gp2c: 0.0.7pl11
- License: GPL v2+
- Programming language: C

5.2. MPC

Participants: Andreas Enge [correspondant], Mickaël Gastineau, Philippe Théveny, Paul Zimmermann [INRIA project-team CARAMEL].

http://mpc.multiprecision.org/.

MPC is a C library for the arithmetic of complex numbers with arbitrarily high precision and correct rounding of the result. It is built upon and follows the same principles as MPFR.

It is a prerequisite for the GNU compiler collection GCC since version 4.5, where it is used in the C and Fortran frontends for constant folding, the evaluation of constant mathematical expressions during the compilation of a program. Since 2011, it is an official GNU project.

- Version: 0.9 Epilobium montanum
- License: LGPL v2.1+
- ACM: G.1.0 (Multiple precision arithmetic)
- AMS: 30.04 Explicit machine computation and programs
- APP: Dépôt APP le 2003-02-05 sous le numéro IDDN FR 001 060029 000 R P 2003 000 10000
- Programming language: C

5.3. MPFRCX

Participant: Andreas Enge.

http://mpfrcx.multiprecision.org/
MPFR CX is a library for the arithmetic of univariate polynomials over arbitrary precision real (MPFR) or complex (MPC) numbers, without control on the rounding. For the time being, only the few functions needed to implement the floating point approach to complex multiplication are implemented. On the other hand, these comprise asymptotically fast multiplication routines such as Toom-Cook and the FFT.

- Version: 0.3.1 Banane
- License: LGPL v2.1+
- Programming language: C

5.4. CM

**Participant:** Andreas Enge.

http://cm.multiprecision.org/

The CM software implements the construction of ring class fields of imaginary quadratic number fields and of elliptic curves with complex multiplication via floating point approximations. It consists of libraries that can be called from within a C program and of executable command line applications. For the implemented algorithms, see [9].

- Version: 0.1 Apfelkraut
- License: LGPL v2+
- Programming language: C

5.5. AVIsogenies

**Participants:** Damien Robert [correspondant], Gaëtan Bisson, Romain Cosset [INRIA project-team CARAMEL].

http://avisogenies.gforge.inria.fr/.

AVISOGENIES (Abelian Varieties and Isogenies) is a MAGMA package for working with abelian varieties, with a particular emphasis on explicit isogeny computation.

Its prominent feature is the computation of \((\ell, \ell)\)-isogenies between Jacobian varieties of genus-two hyper-elliptic curves over finite fields of characteristic coprime to \(\ell\); practical runs have used values of \(\ell\) in the hundreds.

It can also be used to compute endomorphism rings of abelian surfaces, and find complete addition laws on them.

- Version: 0.4
- License: LGPL v2.1+
- Programming language: Magma

5.6. Cubic

**Participant:** Karim Belabas.

http://www.math.u-bordeaux1.fr/~belabas/research/software/cubic-1.2.tgz

CUBIC is a standalone program that prints out generating equations for cubic fields of either signature and bounded discriminant. It depends on the PARI library. The algorithm has quasi-linear time complexity in the size of the output.

- Version: 1.2
- License: GPL v2+
- Programming language: C
ALEA Project-Team (section vide)
5. Software

5.1. RealfluiDS

Participants: Dante de Santis, Gianluca Geraci, Pietro Marco Congedo, Rémi Abgrall [corresponding member].

RealfluiDS is a software dedicated to the simulation of inert or reactive flows. It is also able to simulate multiphase, multimaterial, MHD flows and turbulent flows. There exist 2D and 3D dimensional versions. The 2D version is used to test new ideas that are later implemented in the 3D one. This software implements the more recent residual distribution schemes. The code has been parallelized with and without overlap of the domains. An Uncertainty Quantification library has been added to the software. A partitioning tool exists in the package, which uses Scotch. In the coming years, all the know how of RealfluiDS will be transferred to Aerosol.

5.2. AeroSol

Participants: Damien Genêt [corresponding member], Maxime Mogé, François Pellegrini, Vincent Perrier [corresponding member].

The software AeroSol is jointly developed in the teams Bacchus and Cagire. It is a high order finite element library written in C++. The code design has been carried for being able to perform efficient computations, with continuous and discontinuous finite elements methods on hybrid and possibly curvilinear meshes. The distribution of the unknowns is made with the software PaMPA, developed within the team Bacchus and the team Pumas. Maxime Mogé has been hired on a young engineer position (IJD) obtained in the ADT OuBa HOP for participating to the parallelization of the library, and arrived on November, 1st 2011.

Current features include

- development environment use of CMake for compilation, CTest for automatic tests and memory checking, lcov and gcov for code coverage reports.
- In/Out link with the XML library for handling with parameter files. Reader for GMSH, and writer on the VTK-ASCII legacy format.
- Quadrature formula up to 11th order for Lines, Quadrangles, Hexaeadra, Pyramids, Prisms, up to 14th order for tetrahedron, up to 21st order for triangles.
- Finite elements up to fourth degree for Lagrange finite elements on lines, triangles and quadrangles.
- Geometry elementary geometrical functions for first order lines, triangles, quadrangles.
- Time iteration explicit Runge-Kutta up to fourth order, explicit Strong Stability Preserving schemes up to third order.
- Linear Solvers link with the external linear solver UMFPack.
- Memory handling discontinuous and continuous discretizations based on PaMPA for triangular and quadrangular meshes.
- Numerical schemes continuous Galerkin method for the Laplace problem (up to fifth order) with non consistent time iteration or with direct matrix inversion. Scalar stabilized residual distribution schemes with explicit Euler time iteration have been implemented for steady problems.

5.3. SLOWS

Participant: Mario Ricchiuto [corresponding member].
SLOWs (Shallow-water FLowS) is a C-platform allowing the simulation of free surface shallow water flows with friction. Arbitrary bathymetries are allowed, defined either by some complex piecewise analytical expression, or by $xy-z$ data files, the classical Manning model for friction is used, and an Exner model is implemented for sediment transport. The equations are discretized with a residual based approach which is an adaptation of the schemes developed for aeronautics applications. Due to the inherent unsteadiness of these flows, the time discretization plays an important role. Three different approaches are available, based on conditionally depth-positivity preserving implicit schemes, or on conditionally depth-positivity preserving genuinely explicit discretizations, or on an unconditionally depth-positivity preserving space-time approach.

5.4. COCA

**Participants:** Mario Ricchiuto [corresponding member], Gérard Vignoles.

COCA (CodeOxydationCompositesAutocicatrisants) is a fortran-90 code for the simulation of the oxidation process in self-healing composite materials, developed in collaboration with the Laboratoire des Composites ThermoStructuraux in Bordeaux (UMR-5801 LCTS). This process involves the chemical oxidation of some of the matrix components of the composite, and the production of a liquid oxide that flows and fills material cracks, acting as a diffusion barrier against oxygen and thus protecting the ceramic fibers of the material. COCA simulates this process using a finite element discretization of the model equations. In its current version only transverse cracks are available. COCA makes use of PaStiX to solve the algebraic systems arising from the discretization.

5.5. PaStiX

**Participant:** Pierre Ramet [corresponding member].

This work is supported by the French “Commissariat à l’Énergie Atomique CEA/CESTA” in the context of structural mechanics and electromagnetism applications.

PaStiX ([http://pastix.gforge.inria.fr](http://pastix.gforge.inria.fr)) (Parallel Sparse matrix package) is a scientific library that provides a high performance parallel solver for very large sparse linear systems based on block direct and block ILU(k) iterative methods. Numerical algorithms are implemented in single or double precision (real or complex): LLt (Cholesky), LDLt (Crout) and LU with static pivoting (for non symmetric matrices having a symmetric pattern). This latter version is now used in RealfluiDS (see Section 5.1). The PaStiX library is released under INRIA CeCILL licence.

The PaStiX library uses the graph partitioning and sparse matrix block ordering package Scotch (see Section 5.7). PaStiX is based on an efficient static scheduling and memory manager, in order to solve 3D problems with more than 50 million of unknowns. The mapping and scheduling algorithm handles a combination of 1D and 2D block distributions. This algorithm computes an efficient static scheduling of the block computations for our supernodal parallel solver which uses a local aggregation of contribution blocks. This can be done by taking into account very precisely the computational costs of the BLAS 3 primitives, the communication costs and the cost of local aggregations. We also improved this static computation and communication scheduling algorithm to anticipate the sending of partially aggregated blocks, in order to free memory dynamically. By doing this, we are able to reduce the aggregated memory overhead, while keeping good performance.

Another important point is that our study is suitable for any heterogeneous parallel/distributed architecture when its performance is predictable, such as clusters of multicore nodes. In particular, we now offer a high performance version with a low memory overhead for multicore node architectures, which fully exploits the advantage of shared memory by using an hybrid MPI-thread implementation.

Direct methods are numerically robust methods, but the very large three dimensional problems may lead to systems that would require a huge amount of memory despite any memory optimization. A studied approach consists in defining an adaptive blockwise incomplete factorization that is much more accurate (and numerically more robust) than the scalar incomplete factorizations commonly used to precondition iterative solvers. Such incomplete factorization can take advantage of the latest breakthroughs in sparse direct methods and particularly should be very competitive in CPU time (effective power used from processors and good scalability) while avoiding the memory limitation encountered by direct methods.
5.6. HIPS

**Participant:** Pierre Ramet [corresponding member].

HIPS (Hierarchical Iterative Parallel Solver) is a scientific library that provides an efficient parallel iterative solver for very large sparse linear systems.

The key point of the methods implemented in HIPS is to define an ordering and a partition of the unknowns that relies on a form of nested dissection ordering in which cross points in the separators play a special role (Hierarchical Interface Decomposition ordering). The subgraphs obtained by nested dissection correspond to the unknowns that are eliminated using a direct method and the Schur complement system on the remaining of the unknowns (that correspond to the interface between the sub-graphs viewed as sub-domains) is solved using an iterative method (GMRES or Conjugate Gradient at the time being). This special ordering and partitioning allows for the use of dense block algorithms both in the direct and iterative part of the solver and provides a high degree of parallelism to these algorithms. The code provides a hybrid method which blends direct and iterative solvers. HIPS exploits the partitioning and multistage ILU techniques to enable a highly parallel scheme where several subdomains can be assigned to the same process. It also provides a scalar preconditioner based on the multistage ILUT factorization.

HIPS can be used as a standalone program that reads a sparse linear system from a file; it also provides an interface to be called from any C, C++ or Fortran code. It handles symmetric, unsymmetric, real or complex matrices. Thus, HIPS is a software library that provides several methods to build an efficient preconditioner in almost all situations.

Since August 2008, HIPS is publicly available at [http://hips.gforge.inria.fr](http://hips.gforge.inria.fr) under the INRIA CeCILL licence.

5.7. Scotch

**Participant:** François Pellegrini [corresponding member].

Scotch ([http://www.labri.fr/~pelegrin/scotch/](http://www.labri.fr/~pelegrin/scotch/)) is a software package for parallel and sequential sparse matrix ordering, parallel and sequential graph partitioning, as well as sequential static mapping, and mesh and hypergraph partitioning.

The initial purpose of Scotch was to compute high-quality partitions and static mappings of valued graphs representing parallel computations and target architectures of arbitrary topologies. The original contribution consisted in developing a “divide and conquer” algorithm in which processes are recursively mapped onto processors by using graph bisection algorithms that are applied both to the process graph and to the architecture graph. This allows the mapper to take into account the topology and heterogeneity of the valued graph which models the interconnection network and its resources (processor speed, link bandwidth). As new multicore, multinode parallel machines tend to be less uniform in terms of memory latency and communication bandwidth, this feature is gaining interest.

The software has then been extended in order to produce vertex separators instead of edge separators, using a multilevel framework. Recursive vertex separation is used to compute orderings of the unknowns of large sparse linear systems, which both preserve sparsity when factorizing the matrix and exhibit concurrency for computing and solving the factored matrix in parallel.

Version 5.0 of Scotch, released on August 2007, was the first version to comprise parallel routines. This extension, called PT-Scotch (for “Parallel Threaded Scotch”), is based on a distributed memory model, and makes use of the MPI and, optionally, Posix thread APIs. A distributed graph structure has been defined, which allows users to reserve vertex indices on each processor for future local adaptive refinement. Its parallel graph ordering routine provides orderings which are of the same quality as the ones yielded by the sequential Scotch ordering routine, while competing software ParMETIS experiences a severe loss of quality when the number of processors increase. Scotch5.0 was released under the CeCILL-C free/libre software license, and has been registered at APP (“Agence pour la Protection des Programmes”).
Version 5.1 of Scotch, released on September 2008, extended the parallel features of PT-Scotch, which can now compute graph partitions in parallel by means of a parallel recursive bipartitioning framework. Release 5.1.10 had made Scotch the first full 64-bit implementation of a general purpose graph partitioner, so that PT-Scotch has been able to successfully break the “32-bit” barrier and partition a graph above 2 billion vertices, spread across 2048 processors, at the French CCRT computer center.

Version 6.0, about to be released, offers new sequential features: static mapping with fixed vertices, static remapping, and static remapping with fixed vertices.

Scotch has been integrated in numerous third-party software, which indirectly contribute to its diffusion. For instance, it is used by the ZOLTAN module of the Trilinos software (SANDIA Labs), by Code_Aster Libre, a GPLed thermal and mechanical analysis software developed by French state-owned electricity producer EDF, by the parallel solvers MUMPS (ENSEEITH/IRIT, LIP and LaBRI), SuperLUdist (U.C. Berkeley), PaStiX (LaBRI) and HIPS (LaBRI), as well as by several other scientific computing software.

5.8. MMG3D

**Participant:** Cécile Dobrzynski [corresponding member].

MMG3D is a tetrahedral fully automatic remesher. Starting from a tetrahedral mesh, it produces quasi-uniform meshes with respect to a metric tensor field. This tensor prescribes a length and a direction for the edges, so that the resulting meshes will be anisotropic. The software is based on local mesh modifications and an anisotropic version of Delaunay kernel is implemented to insert vertices in the mesh. Moreover, MMG3D allows one to deal with rigid body motion and moving meshes. When a displacement is prescribed on a part of the boundary, a final mesh is generated such that the surface points will be moved according this displacement. MMG3D is used in particular in GAMMA for their mesh adaptation developments, but also at EPFL (maths department), Dassault Aviation, Lemma (a french SME), etc. MMG3D can be used in FreeFem++ (http://www.freefem.org), a free software which eases the solving of PDEs. More details can be found on http://www.math.u-bordeaux1.fr/~dobj/logiciels/mmg3d.php.

5.9. Montjoie

**Participant:** Marc Duruflé [corresponding member].

Montjoie is a finite element code initially handling only quadrilateral/hexaedral elements. Because of the tensorization of these elements, efficient algorithms can be written for the computation of finite element matrices. It can handle tetrahedra, prisms, pyramids, hexaedra with continuous finite element, edge elements and discontinuous Galerkin formulations. A local order of approximation can be used in each element of the mesh.

5.10. PLATO

**Participants:** Hervé Guillard [PUMAS], Laure Combe [PUMAS,contact], Cédric Lachat, Pierre Ramet [corresponding member].

The development of PLATO (A platform for Tokamak simulation) (http://www-sop.inria.fr/pumas/plato.php) is being supported by an ADT action of the D2T. PLATO is a suite of data and softwares dedicated to the geometry and physics of Tokamaks and its main objective is to provide the Inria large scale initiative “FUSION” teams working with plasma fluid models with a common development tool. The construction of this platform will integrate the following developments.

1. A (small) database corresponding to axi-symmetrical solutions of the equilibrium plasma equations for realistic geometrical and magnetic configurations (ToreSupra, JET and ITER). The construction of meshes is always an important time consuming task. Plato will provide meshes and solutions corresponding to equilibrium solutions that will be used as initial data for more complex computations.
2. A set of tool for the handling, manipulation and transformation of meshes and solutions using different discretisations (P1, Q1, P3, etc)
3. Numerical templates allowing the use of 3D discretization schemes using finite element schemes in the poloidal plane and spectral Fourier or structured finite volume representations in the toroidal one.

4. Several applications (Ideal MHD and drift approximation) used in the framework of the Inria large scale initiative “FUSION”.

5.11. PaMPA

**Participants:** Cécile Dobrzynski, Hervé Guillard [PUMAS], Laurent Hascoët [Tropics], Cédric Lachat, François Pellegrini [Corresponding member].

PaMPA (“Parallel Mesh Partitioning and Adaptation”) is a middleware library dedicated to the management of distributed meshes. Its purpose is to relieve solver writers from the tedious and error prone task of writing again and again service routines for mesh handling, data communication and exchange, remeshing, and data redistribution. It is based on a distributed data structure that represents meshes as a set of entities (elements, faces, edges, nodes, etc.), linked by relations (that is, computation dependencies).

Version 0.1 allows users to declare a distributed mesh, declare values attached to the entities of the meshes (e.g. temperature attached to elements, pressures to the faces, etc.), exchange values between overlapping entities located at the boundaries of subdomains assigned to different processors, and iterate over the relations of entities (e.g. iterate over the faces of elements).

PaMPA is already used as the data structure manager for two solvers being developed at INRIA: Plato and Aerosol.

PaMPA will soon interface with Scotch for mesh redistribution, and with MMG3D to offer parallel remeshing features (in this particular example, for tetrahedral elements).
5. Software

5.1. AeroSol

**Participants:** Damien Genet [Bacchus], Maxime Mogé, Francois Pellegrini [Bacchus], Vincent Perrier [correspondant].

The software AeroSol is jointly developed in the team Bacchus and the team Cagire. It is a high order finite element library written in C++. The code design has been carried for being able to perform efficient computations, with continuous and discontinuous finite elements methods on hybrid and possibly curvilinear meshes. The distribution of the unknowns is made with the software PaMPA, developed within the team Bacchus and the team Pumas. Maxime Mogé has been hired on a young engineer position (IJD) obtained in the ADT OuBa HOP for participating to the parallelization of the library, and arrived on November, 1st 2011.

Current features include

- **development environement** use of CMake for compilation, CTest for automatic tests and memory checking, lcov and gcov for code coverage reports.
- **In/Out** link with the XML library for handling with parameter files. Reader for GMSH, and writer on the VTK-ASCII legacy format.
- **Quadrature formula** up to 11th order for Lines, Quadrangles, Hexaedra, Pyramids, Prisms, up to 14th order for tetrahedron, up to 21st order for triangles.
- **Finite elements** up to fourth degree for Lagrange finite elements on lines, triangles and quadrangles.
- **Geometry** elementary geometrical functions for first order lines, triangles, quadrangles.
- **Time iteration** explicit Runge-Kutta up to fourth order, explicit Strong Stability Preserving schemes up to third order.
- **Linear Solvers** link with the external linear solver UMFPack.
- **Memory handling** discontinuous and continuous discretizations based on PaMPA for triangular and quadrangular meshes.
- **Numerical schemes** continuous Galerkin method for the Laplace problem (up to fifth order) with non consistent time iteration or with direct matrix inversion. Scalar stabilized residual distribution schemes with explicit Euler time iteration have been implemented for steady problems.
5. Software

5.1. C++ library Concha

Participants: Roland Becker, Daniela Capatina, Robert Luce, David Trujillo.

The objectives of our library CONCHA are to offer a flexible and extensible software with respect to:

- Numerical methods and
- Physical models.

The aim is to have a flexible code which could easily switch between the different discretizations, in order to provide a toolbox for rapid testing of new ideas.

The software architecture is designed in such a way that a group of core developers can contribute in an efficient manner, and that independent development of different physical applications is possible. Further, in order to accelerate the integration of new members and in order to provide a basis for our educational purposes (see Section 9.3), the software proposes different entrance levels. The basic structure consists of a common block, and several special libraries which correspond to the different fields of applications described in Sections 4.1 – 4.4 Hyperbolic solvers, Low-Mach number flow solvers, DNS, and viscoelastic flows. A more detailed description of each special library may be found below. In order to coordinate the cooperative development of the library, Concha is based on the INRIA-Gforge.

5.2. User interface and python interface

Participants: Roland Becker, David Trujillo.

A graphical user-interface facilitate the use of the C++-library. It has been developed by Guillaume Baty (former technical staff) in collaboration with Pierre Puiseux (associate professor at LMAP). All members of the team have been involved in the testing of the interface. The first objective is to provide an easy way of installation and to facilitate the usage. To this end we use the python language with Qt in order to take advantage of higher level libraries, which allow us to reduce development time.

![Figure 5. Graphical user interface: option panel (left) and process panel (right) of the install tool.](image-url)
We are confronted with heterogenous backgrounds and levels of implication of the developers and users. It seems therefore crucial to be able to respond to the different needs. Our aim is to facilitate the development of the library, and at the same time, to make it possible that our colleagues involved in physical modeling can have access to the functionality of the software with a reasonable investment of time. Two graphical user interfaces have been developed: one for the installation of the library and another one for the building and execution of projects. They are based on common database and scripts written in python. The scripts can also be launched in a shell. In Figure 5 the user interface of the install tool is shown. The option panel allows to choose the components for conditional compilation and the compilation type (debug and release).

In Figure 6 the user interface of the project tool is shown. A project consists of a number of sources files and a parameter file used by the C++-executable. The sources define classes derived from the library, which are used to specify certain data such as boundary conditions and employed finite element spaces. The parameter file contains algorithmic information and physical parameters. It is generated from a database by the python utilities.

The tools offered by this development platform are based on a python interface for the library, called pyConcha. It offers a common interface, based on a plugin-system, which allows the development of command line tools in parallel. This year the consolidation of the interface part of pyConcha has been an important task. The pyConcha library is now a framework rather than a simple interface to Concha C++ library. It allows now creation of plugins, so that each user-programmer can customize pyConcha to his own goals. Previously, two main programs where working: concha-install.py to install library, and concha-project.py for (semi-)end-users. Both are now plugins of pyConcha, and can be launched by pyConcha at startup. A plugin visualization could now be developed in an independant way, and launched by pyConcha on demand.

The structure of pyConcha framework is clearly splitted in various modules(layers): Command Line Interface module, Graphical User Interface module and Handlers modules, see Figure 7. A great effort has been made for internationalization of pyConcha.

5.3. Parallelization

Participants: Roland Becker, Elies Bergounioux, David Trujillo.
The parallelization of the library is done in collaboration with the INRIA-team Runtime, Marie-Christine Counilh and Olivier Aumage and with Séphanie Delage (CRI, UPPA). Elies Bergounioux worked on this topic and was financed as an IID by the ADT AMPLI. The strategy for the parallelization is based on a hybrid approach using OpenMP and MPI.

5.4. Euler equations

Participants: Roland Becker, Robert Luce, Eric Schall, David Trujillo.

Based on the library CONCHA, we have develop a solver for hyperbolic PDE’s based on DGFEM. So far different standard solvers for the Euler equations such as Lax-Friedrichs, Steger-Warming, and HLL have been implemented for test problems. A typical example is the scram jet test case shown in Figure 8.

5.5. Incompressible flow solvers

Participants: Roland Becker, Daniela Capatina, Robert Luce, David Trujillo.

We have started the validation of the implementation of different finite element methods for incompressible flows at hand of standard benchmark problems as the Stokes flow around a symmetric cylinder [65] and the stationary flow around a slightly non symmetric cylinder [74], see Figure 9.

5.6. DNS

Participants: Roland Becker, David Trujillo, Elies Bergounioux.

For the direct numerical simulation of incompressible turbulent flows, we have started to develop a special solver based on structured meshes with a fast multigrid algorithm incorporating projection-like schemes. The main idea is to use non-conforming finite elements for the velocities with piecewise constant pressures, leading to a special structure of the discrete Schur complement, when an explicit treatment of the convection and diffusion term is used.
Figure 8. Computed Mach-number distribution for the Scramjet test problem.

Figure 9. Flow fields for the Stokes (above) and Navier-Stokes (below) benchmark.
5.7. Polymer flow

Participants: Roland Becker, Daniela Capatina, Julie Joie, Didier Graebling.

Based on our library CONCHA we have implemented a three-field formulation with unknowns \((u, p, \tau)\) for the two-dimensional Navier-Stokes equations, based on nonconforming finite elements. The extension to the Giesekus-model for polymers has been achieved, see Section 6.7. In the case of Newtonian flows, the extra-tensor can be eliminated in order to reduce storage and computing time. This procedure serves as a preconditioner in the general case. The aim is to provide software tools for the problems in Section Viscoelastic flows.

5.8. Validation and comparison with other CFD-software

Participants: Daniela Capatina, Didier Graebling, Julie Joie, Eric Schall.

We intend to compare computations based on CONCHA with other codes at hand of the prototypical test problems described above. This allows us to evaluate the potential of our numerical schemes concerning accuracy, computing time and other practical expects such as integration with mesh generators and post-processing. At the same time, this, unfortunately very time-consuming, benchmarking activity allows us to validate our own library. The following commercial and research tools might be considered: Aéro3d (INRIA-Smash), AVBP (CERFACS), ELSA (ONERA), Fluent (ANSYS), and OpenFOAM (OpenCfd), and Polyflow® (ANSYS). So far, we have compared our code for the Giesekus model of polymer flows with the commercial software Polyflow®, see Section 4.2.
5. Software

5.1. Package “ClustOfVar”

This R package is dedicated to cluster analysis of a set of variables. Variables can be quantitative, qualitative or a mixture of both. A new version of the package is available via the link http://cran.r-project.org/web/packages/ClustOfVar/index.html since novembre 2011. The new version improves the computational time of the "kmeansvar" function used for k-means type clustering of variables. This function is now able to deal with datasets of several thousands of variables like genomic data. The package is detailed in a paper submitted for publication [44]. It has been presented in several conferences [34], [35].

5.2. Package “PCAmixdata”

This package is dedicated to factorial analysis and rotation of quantitative data, qualitative data, or mixed data. The PCAMIX method, proposed in this package includes the ordinary principal component analysis (PCA) and multiple correspondence analysis (MCA) as special cases. Orthogonal varimax rotation of the principal components of PCAMIX is also implemented in this package. Theoretical and practical results about the new rotation algorithm available in the package is in revision for publication [45] and has been presented in [36].
GEOSTAT Project-Team

5. Software

5.1. Software

Participants: Hussein Yahia [correspondant], Antonio Turiel, Joel Sudre.

FluidExponents: software implementation of the MMF, written in Java, in a cooperative development mode on the INRIA GForge, deposited at APP in 2010. Contact: hussein.yahia@inria.fr. FluidExponents implements nonlinear signal processing on various types of input data (including NETCDF).

- Version: 0.8
5. Software

5.1. eLYSe

**Participants:** Olivier Saut [correspondant], Raphael Bahegne, Vincent Hubert, Jean-Baptiste Lagaert, Mathieu Specklin.

eLYse is a numerical platform used for our computations in Biology (tumor growth), micro-fluidics and complex Newtonian fluid flows. The platform is divided in two libraries: one is devoted to the modelling equations and the other one includes the numerical solvers. For example, we are able to treat (in 2D and 3D) transport equations, diffusion equations, Navier-Stokes equations, Maxwell system and the interaction fluid-structure by level-set and penalization methods. The solvers are based on finite volume methods on cartesian grids and allow parallel computations. See also the web page [http://www.math.u-bordeaux1.fr/~saut/wp/?page_id=201](http://www.math.u-bordeaux1.fr/~saut/wp/?page_id=201).

- Version: 0.4
- ACM: ACM J.2 J.3 G.1.8 G.1.10
- AMS: AMS65Z05 35Q92
- Keywords: Modélization and numerical simulations, Finite volume methods, Level Set approach, Penalization method
- APP: En cours
- Type of human computer interaction: console
- OS/Middleware: Platform developed on Mac OS X architecture.
- Programming language: C++
- Documentation: doxygen.

5.2. Kesaco

**Participants:** Olivier Saut [correspondant], Raphaël Bahègne, Damiano Lombardi, Mathieu Specklin.

Kesaco is a set of libraries and programs aiming at applications of mathematical modeling in clinical oncology. It features:

- A library of specialized mathematical model describing the growth of different types of cancers (secondary tumors in the lung, gliomas).
- A set of programs useful to validate mathematical models (compute the various behavior they can produce) and to build databases of numerical simulations.
- Segmentation and registration routines to use medical images directly in our numerical codes.
- Calibration methods to recover the parameters of the models using sequences of medical images. Three techniques are implemented (a genetic algorithm, a technique based on reduced order models, a sensitivity technique).
All these routines are adapted to run on a MP architecture. The webpage may be found at [http://www.math.u-bordeaux1.fr/~saut/wp/?page_id=345](http://www.math.u-bordeaux1.fr/~saut/wp/?page_id=345).

- Version: 0.1
- Keywords: Modélisation and numerical simulations
- APP: En cours
- Type of human computer interaction: console
- OS/Middleware: Platform developed on Mac OS X architecture.
- Required library or software: eLYSe, Insight Toolkit ([http://www.itk.org](http://www.itk.org))
- Programming language: C++
- Documentation: doxygen.

### 5.3. NaSCar

**Participant:** Michel Bergmann [correspondant].

This code is devoted to solve 3D-flows in around moving and deformable bodies. The incompressible Navier-Stokes equations are solved on fixed grids, and the bodies are taken into account thanks to penalization and/or immersed boundary methods. The interface between the fluid and the bodies is tracked with a level set function or in a Lagrangian way. The numerical code is fully second order (time and space). The numerical method is based on projection schemes of Chorin-Temam’s type. The code is written in C language and use Petsc ([http://www.mcs.anl.gov/petsc/petsc-as/](http://www.mcs.anl.gov/petsc/petsc-as/)) library for the resolution of large linear systems in parallel.

NaSCar can be used to simulate both hydrodynamic bio-locomation as fish like swimming and aerodynamic flows such wake generated by a wind turbine.

- Version: 1
- Keywords: numerical analyse, fluid mechanics, langage C, PETSc
- Software benefit: simulate a flow around a deformable obstacle, moving into a fluid.
- APP: En cours
- Patent: non
- Type of human computer interaction: human for the moment
- OS/Middleware: unix, linux, mac os
- Required library or software: PETSc item Programming language: C
- Documentation: in progress

### 5.4. Other MC2 codes

- Penalization techniques on cartesian grids to solve incompressible Navier-Stokes equations
  - Vortex: sequential, Vortex In-Cell (VIC) scheme: hybrid vortex methods based on the combination of Lagrangian mesh-free schemes and Eulerian grid based schemes on the same flow region.
  - NS2D(3D): DNS, Finite Difference scheme, Multrid solver, parallel MPI.

- Unstructured body fitted meshes
  - Richards: 2D Unstructured finite element code, implicit solver, sequential, to solve the transport-diffusion equations through a porous media including tidal forcing and mechanisms of diagenesis.
  - development inside FluidBox software in collaboration with BACCHUS. 2D-3D unstructured meshes, Stabilized Finite Elements method (SUPG), RANS turbulence model, parallel: Domain Decomposition and MPI.

- Compressible flows and elliptic problems
  - Compressible flows: 2D-3D finite volume scheme pour compressible Euler/NS equations on cartesian grids
  - Elliptic problems: 2D-3D finite difference scheme for elliptic interface problems, sequential and parallel
5. Software

5.1. BaPCod – a generic Branch-and-Price Code

Participants: Romain Leguay [Software Engineer], Pierre Pesneau, Ruslan Sadykov, François Vanderbeck [correspondant].

BaPCod is a prototype code that solves Mixed Integer Programs (MIP) by application of a Dantzig-Wolfe reformulation technique. The reformulated problem is solved using a branch-and-price (column generation) algorithm. This software platform, made of C++ classes, offers a “black-box” implementation that does not require user input and is not application specific. The features are

(i) the automation of the Dantzig-Wolfe reformulation process (the user defines a mixed integer programming problem in terms of variables and constraints, identifies subproblems, and can provide the associated solvers if available, but he does not need to explicitly define the reformulation, the explicit form of the columns, their reduced cost, or the Lagrangian bounds).

(ii) a default column generation procedure with standard initialization and stabilization (it may offer a selection of solvers for the master) – the issue of stabilization is discussed in [1], and

(iii) a default branching scheme – recent progress has been made on the issue of generic branching scheme in [23].

(iv) default primal heuristics specially developed for use in a decomposition framework [58].

The prototype software was/is used as background solver for 5 PhD thesis. It also served as the framework for our comparative study in a INRIA collaborative research action [1]. It has been experimented by two of our industrial partners, Exeo Solutions (Bayonne), on an inventory routing problem, and Orange Lab (France Telecom, Paris) on network design problems, time tabling problem by EURODECISION and it is currently being tested by the University Paris 6 and EDF. The prototype also enables us to be very responsive in our industrial contact.

See also the web page https://wiki.bordeaux.inria.fr/realopt/pmwiki.php/Project/BaPCod.

• Version: 1
ANUBIS Project-Team (section vide)
CARMEN Team (section vide)
5. Software

5.1. SPECFEM3D

The MAGIQUE-3D project is based (in part) on existing software packages, which are already validated, portable and robust. The SPECFEM3D software package, developed by Dimitri Komatitsch and his colleagues in collaboration with Jeroen Tromp and his colleagues at the California Institute of Technology and at Princeton University (USA), and which is still actively maintained by Dimitri Komatitsch and his colleagues, allows the precise modeling of seismic wave propagation in complex three-dimensional geological models. Phenomena such as anisotropy, attenuation (i.e., anelasticity), fluid-solid interfaces, rotation, self-gravitation, as well as crustal and mantle models can be taken into account. The software is written in Fortran95 with MPI message-passing on parallel machines. It won the Gordon Bell Prize for best performance of the Supercomputing’2003 conference. In 2006, Dimitri Komatitsch established a new collaboration with the Barcelona Supercomputing Center (Spain) to work on further optimizing the source code to prepare it for very large runs on future petaflops machines to solve either direct or inverse problems in seismology. Optimizations have focused on improving load balancing, reducing the number of cache misses and switching from blocking to non-blocking MPI communications to improve performance on very large systems. Because of its flexibility and portability, the code has been run successfully on a large number of platforms and is used by more than 150 academic institutions around the world. In November 2008 this software package was again among the six finalists of the prestigious Gordon Bell Prize of the SuperComputing’2008 conference in the USA [50] for a calculation performed in parallel on 150,000 processor cores, reaching a sustained performance level of 0.16 petaflops.

5.2. Hou10ni

This software, written in FORTRAN 90, simulates the propagation of acoustic waves in heterogeneous 2D and 3D media. It is based on an Interior Penalty Discontinuous Galerkin Method (IPDGM). The 2D version of the code has been implemented in the Reverse Time Migration (RTM) software of TOTAL in the framework of the PhD. thesis of Caroline Baldassari and the 3D version should be implemented soon. The 2D code allows for the use of meshes composed of cells of various order ($p$-adaptivity in space). For the time discretization, we used the local time stepping strategy described at section 3.2, item High-Order Schemes in Space and Time which permits not only the use of different time-step, but also to adapt the order of the time-discretization to the order of each cells ($hp$-adaptivity in time). These functionalities will be soon implemented in the 3D code.

The main competitors of Hou10ni are codes based on Finite Differences, Spectral Element Method or other Discontinuous Galerkin Methods (such as the ADER schemes). During her PhD. thesis, Caroline Baldassari compared the solution obtained by Hou10ni to the solution obtained by a Finite Difference Method and by a Spectral Element Method (SPECFEM). To evaluate the accuracy of the solutions, we have compared it to analytical solutions provided by the codes Gar6more (see below). The results of these comparisons is: a) that Hou10ni outperforms the Finite Difference Methods both in terms of accuracy and of computational burden and b) that its performances are similar to Spectral Element Methods. Since Hou10ni allows for the use of meshes based on tetraedrons, which are more appropriate to mesh complex topographies, and for the $p$-adaptivity, we decided to implement it in the RTM code of TOTAL. Of course, we also used these comparisons to validate the code. Now, it remains to compare the performances of Hou10ni to the ADER schemes.

5.3. Gar6more3D

Participants: Julien Diaz [correspondant], Abdelaâziz Ezziani.
This code computes the analytical solution of problems of waves propagation in two layered 3D media such as: acoustic/acoustic- acoustic/elastodynamic- acoustic/porous- porous/porous, based on the Cagniard-de Hoop method.

See also the web page http://web.univ-pau.fr/~jdiaz1/softwares.html.

The main objective of this code is to provide reference solutions in order to validate numerical codes. They have been already used by J. Tromp and C. Morency to validate their code of poroelastic wave propagation [67]. They are freely distributed under a CECILL licence and can be downloaded on the website http://web.univ-pau.fr/~jdiaz1/softwares.html. As far as we know, the main competitor of this code is EX2DELDEL (available on http://www.spice-rtn.org), but this code only deals with 2D acoustic or elastic media. Our codes seem to be the only one able to deal with bilayered poroelastic media and to handle the three dimensional cases.

- ACM: J.2
- AMS: 34B27 35L05 35L15 74F10 74J05
- Programming language: Fortran 90
5. Software

5.1. Inria Bioscience Resources

**Participants:** Olivier Collin [correspondant], Frédéric Cazals, Mireille Régnier, Marie-France Sagot, Hélène Touzet, Hidde de Jong, David Sherman, Marie-Dominique Devignes, Dominique Lavenier.

Inria Bioscience Resources is a portal designed to improve the visibility of bioinformatics tools and resources developed by Inria teams. This portal will help the community of biologists and bioinformaticians understand the variety of bioinformatics projects in Inria, test the different applications, and contact project-teams. Eight project-teams participate in the development of this portal. Inria Bioscience Resources is developed in an Inria Technology Development Action (ADT).

5.2. Magus: Collaborative Genome Annotation

**Participants:** David James Sherman [correspondant], Pascal Durrens, Natalia Golenetskaya, Florian Lajus, Tiphaine Martin.

As part of our contribution the Génolevures Consortium, we have developed over the past few years an efficient set of tools for web-based collaborative annotation of eukaryote genomes. The MAGUS genome annotation system integrates genome sequences and sequences features, *in silico* analyses, and views of external data resources into a familiar user interface requiring only a Web navigator. MAGUS implements the annotation workflows and enforces curation standards to guarantee consistency and integrity. As a novel feature the system provides a workflow for *simultaneous annotation* of related genomes through the use of protein families identified by *in silico* analyses; this has resulted in a three-fold increase in curation speed, compared to one-at-a-time curation of individual genes. This allows us to maintain Génolevures standards of high-quality manual annotation while efficiently using the time of our volunteer curators.

MAGUS is built on: a standard sequence feature database, the Stein lab generic genome browser [55], various biomedical ontologies (http://obo.sf.net), and a web interface implementing a representational state transfer (REST) architecture [35].

For more information see magus.gforge.inria.fr, the MAGUS Gforge web site. MAGUS is developed in an Inria Technology Development Action (ADT).

5.3. YAGA: Yeast Genome Annotation

**Participants:** Pascal Durrens, Tiphaine Martin [correspondant].

With the arrival of new generations of sequencers, laboratories, at a lower cost, can be sequenced groups of genomes. You can no longer manually annotate these genomes. The YAGA software’s objective is to syntactically annotate a raw sequence (genetic element: gene, CDS, tRNA, centromere, gap, ...) and functionally as well as generate EMBL files for publication. The annotation takes into account data from comparative genomics, such as protein family profiles.

After determining the constraints of the annotation, the YAGA software can automatically annotate *de novo* all genomes from their raw sequences. The predictors used by the YAGA software can also take into account the data RNAseq to reinforce the prediction of genes. The current settings of the software are intended for annotation of the genomes of yeast, but the software is adaptable for all types of species.

5.4. BioRica: Multi-scale Stochastic Modeling

**Participants:** David James Sherman [correspondant], Rodrigo Assar Cuevas, Alice Garcia.
BioRica is a high-level modeling framework integrating discrete and continuous multi-scale dynamics within
the same semantics field. A model in BioRica node is hierarchically composed of nodes, which may be existing
models. Individual nodes can be of two types:

- Discrete nodes are composed of states, and transitions described by constrained events, which
can be non deterministic. This captures a range of existing discrete formalisms (Petri nets, finite
automata, etc.). Stochastic behavior can be added by associating the likelihood that an event fires
when activated. Markov chains or Markov decision processes can be concisely described. Timed
behavior is added by defining the delay between an event’s activation and the moment that its
transition occurs.
- Continuous nodes are described by ODE systems, potentially a hybrid system whose internal state
flows continuously while having discrete jumps.

The system has been implemented as a distributable software package

The BioRica compiler reads a specification for hierarchical model and compiles it into an executable simulator.
The modeling language is a stochastic extension to the AltaRica Dataflow language, inspired by work of
Antoine Rauzy. Input parsers for SBML 2 version 4 are currently being validated. The compiled code uses the
Python runtime environment and can be run stand-alone on most systems [36].

For more information see biorica.gforge.inria.fr, the BioRica Gforge web site. BioRica was developed as an
Inria Technology Development Action (ADT).

5.5. Pathtastic: Inference of whole-genome metabolic models

Participants: David James Sherman [correspondant], Pascal Durrens, Nicolás Loira, Anna Zhukova.

Pathtastic is a software tool for inferring whole-genome metabolic models for eukaryote cell factories. It is
based on metabolic scaffolds, abstract descriptions of reactions and pathways on which inferred reactions are
hung are eventually connected by an interactive mapping and specialization process. Scaffold fragments can
be repeatedly used to build specialized subnetworks of the complete model.

Pathtastic uses a consensus procedure to infer reactions from complementary genome comparisons, and an
algebra for assisted manual editing of pathways.

For more information see pathtastic.gforge.inria.fr, the Pathtastic Gforge web site.

5.6. Génolevures On Line: Comparative Genomics of Yeasts

Participants: David James Sherman, Pascal Durrens [correspondant], Natalia Golenetskaya, Tiphaine Martin.

The Génolevures online database provides tools and data for exploring the annotated genome sequences of
more than 20 genomes, determined and manually annotated by the Génolevures Consortium to facilitate
comparative genomic studies of hemiascomycetous yeasts. Data are presented with a focus on relations
between genes and genomes: conservation of genes and gene families, speciation, chromosomal reorganization
and synteny. The Génolevures site includes an area for specific studies by members of its international
community.

Génolevures online uses the MAGUS system for genome navigation, with project-specific extensions devel-
oped by David Sherman, Pascal Durrens, and Tiphaine Martin. An advanced query system for data mining in
Génolevures is being developed by Natalia Golenetskaya. The contents of the knowledge base are expanded
and maintained by the CNRS through GDR 2354 Génolevures. Technical support for Génolevures On Line is
provided the CNRS through UMR 5800 LaBRI.

For more information see genolevures.org, the Génolevures web site.
5. Software

5.1. Hubble

Participants: Ludovic Courtès [correspondant], Nicolas Bonichon.

Hubble is implemented in Scheme, using GNU Guile version 2. Details of the simulation, such as keeping track of processor occupation and network usage, are taken care of by SimGrid, a toolkit for the simulation of distributed applications in heterogeneous distributed environments.

The input to Hubble is an XML description of the DAG of build tasks. For each task, a build duration and the size in bytes of the build output are specified. For our evaluation purposes, we collected this data on a production system, the http://hydra.nixos.org/ build farm hosted at the Technical University of Delft. The DAG itself is the snapshot of the Nix Package Collection (Nixpkgs) corresponding to this data. Hubble has its own in-memory representation of the DAG in the form of a purely functional data structure.

The Nixpkgs DAG contains fixed-output nodes, i.e., nodes whose output is known in advance and does not require any computation. These nodes are typically downloads of source code from external web sites. The raw data collected on http://hydra.nixos.org/ specifies a non-zero duration for these nodes, which represents the time it took to perform the download. This duration info is irrelevant in our context, since they don’t require any computation, and Hubble views these nodes as instantaneous.

See also the web page http://hubble.gforge.inria.fr/.


5.2. namdP2P

Participants: Nicolas Bonichon [correspondant], Olivier Beaumont.

NamdP2P is a distributed implementation of ABF method using NAMD. It is worth noting that NAMD is designed to run on high-end parallel platforms or clusters, but not to run efficiently on instable and distributed platforms.


5.3. Malleable minWCT

Participants: Nicolas Bonichon [correspondant], Olivier Beaumont, Lionel Eyraud-Dubois.

This applet considers the scheduling of malleable tasks with bounded amount of processing resources. The goal is to compute schedules that minimize the weighted completion time of tasks. The applet generates all possible greedy schedules for a given instance and displays only the best ones.

This applet illustrates the complexity of finding an optimal order of tasks [31].

See also the web page http://www.labri.fr/perso/bonichon/malleable.


5.4. Bedibe

Participants: Lionel Eyraud-Dubois [correspondant], Przemyslaw Uznanski.

Bedibe (Benchmarking Distributed Bandwidth Estimation) is a software to compare different models for bandwidth estimation on the Internet, and their associated instantiation algorithms. The goal is to ease the development of new models and algorithms, and the comparison with existing solutions. The development of this software is just starting.
See also the web page  http://bedibe.gforge.inria.fr/ .

5.5. MineWithRounds

Participants: Sofian Maabout [correspondant], Nicolas Hanusse.

This software extracts the Maximal Frequent Itemsets from a transaction data base. It is designed in C++ using OpenMP Library to take full advantage of multicore, multi-cpu machines.
5. Software

5.1. Introduction

We describe in this section the software that we are developing. The first two (MaPHyS and EPSN) will be
the main milestones of our project. The other software developments will be conducted in collaboration with
academic partners or in collaboration with some industrial partners in the context of their private R&D or
production activities. For all these software developments, we will use first the various (very) large parallel
platforms available through CERFACS and GENCI in France (CCRT, CINES and IDRIS Computational
Centers), and next the high-end parallel platforms that will be available via European and US initiatives or
projects such that PRACE.

5.2. MaPHyS

MaPHyS (Massively Parallel Hybrid Solver) is a software package whose protpotype was initially developed
in the framework of the PhD thesis of Azzam Haidar (CERFACS) and futher consolidated thanks to the ANR-
CIS Solstice funding. This parallel linear solver couples direct and iterative approaches. The underlying idea
is to apply to general unstructured linear systems domain decomposition ideas developed for the solution of
linear systems arising from PDEs. The interface problem, associated with the so called Schur complement
system, is solved using a block preconditioner with overlap between the blocks that is referred to as Algebraic
Additive Schwarz.

In the framework of the INRIA technologic development actions; 24 man-month engineer (Yohan Lee-Tin-
Yien) have been allocated to this software activity for the 2009-2011 period. The initial software prototype has
been completly redesigned in order to enable us to easily interface any sparse direct solvers and develop new
preconditioning technique. The first public release of the software is planned eraly 2012. The same software
effort has been undertaken for interfacing any graph partitioning tools.

The MaPHyS package is very much a first outcome of the research activity described in Section 3.3 . Finally,
MaPHyS is a preconditioner that can be used to speed-up the convergence of any Krylov subspace method. We
forsee to either embed in MaPHyS some Krylov solvers or to release them as standalone packages, in particular
for the block variants that will be some outcome of the studies discussed in Section 3.3 .

5.3. EPSN

EPSN (Environment for Computational Steering) is a software environment for the steering of legacy parallel-
distributed simulations with simple GUI or more complex (possibly parallel) visualization programs (see
Figure 1 ). In order to make a legacy simulation steerable, the user annotates the sourcecode with the EPSN
API. These annotations provide the EPSN environment with two kinds of information: the description of the
program structure according to a Hierarchical Task Model (HTM) and the description of the distributed data
that will be remotely accessible. EPSN provides a distributed data model, that handles common scientific
objects such as parameters, structured grids, particles/atoms and unstructured meshes. It is then possible to
dynamically connect EPSN with a client program, that provides a GUI with some visualization & interaction
features, as for instance SIMONE (Simulation MONitoring for Epsn). Once a client is connected, it interacts
with the simulation via EPSN API. It is possible : 1) to control the execution flow of the remote simulation; 2)
to access/modify its data onthefly; and 3) finally to invoke advanced user-defined routines in the simulation.
The current version of EPSN is fully based on CORBA for communication on heterogeneous system and
VTK/Paraview for visualization. A new release of EPSN, that will be fully based on MPI to handle efficient
communication, is currently under development. A prototype is already working.
Figure 1. EPSN: software environment for $M \times N$ computational steering.
5.4. MPICPL

MPICPL (MPI CouPLing) is a software library dedicated to the coupling of parallel legacy codes, that are based on the well-known MPI standard. It proposes a lightweight and comprehensive programming interface that simplifies the coupling of several MPI codes (2, 3 or more). MPICPL facilitates the deployment of these codes thanks to the mpicplrun tool and it interconnects them automatically through standard MPI inter-communicators. Moreover, it generates the universe communicator, that merges the world communicators of all coupled-codes. The coupling infrastructure is described by a simple XML file, that is just loaded by the mpicplrun tool. Future releases will incorporate new features for checkpoint/restart and dynamic parallel code connection.

MPICPL was developed by the Inria HiePACS project-team for the purpose of the ANR CIS NOSSI. It uses advanced features of MPI2 standard. The framework is publicly available at Inria Gforge: http://mpicpl.gforge.inria.fr.

5.5. MONIQA

MONIQA (MONitoring graphic user Interface for Qm/mm Applications) is a GUI specially designed for the monitoring & steering of the QM/MM application in the ANR CIS NOSSI project. It is based on Tulip, a graph visualization software http://tulip.labri.fr), used to display atoms and molecules. It proposes two working modes: offline or online. The offline mode is mainly used to load input files of DL_POLY & Siesta, and to prepare the quantum region for the QM/MM coupling. In online mode, the end-user can monitor & interact with the running QM/MM application thanks to EPSN. It is thus possible to visualize molecular and physical data (distances, angles, charges, energies), and to change simulation parameters on-the-fly, such as the target temperature of the system, thermo or barostat parameters, verbosity of output, ... MONIQA is based on QT4. It was developed specifically for the ANR NOSSI project and is available (restricted access) at Inria Gforge: http://nossi.gforge.inria.fr.

5.6. ScalFMM

ScalFMM (Parallel Fast Multipole Library for Large Scale Simulations) is a software library to simulate N-body interactions using the Fast Multipole Method. ScalFMM is based on the FMB prototype developed by Pierre Fortin during his PhD thesis.

In the framework of the INRIA technologic development actions; 24 man-month engineer (Bérenger Bramas) have been allocated to this software activity started in January 2011.

ScalFMM intends to offer all the functionalities needed to perform large parallel simulations while enabling an easy customization of the simulation components: kernels, particles and cells. It works in parallel in a shared/distributed memory model using OpenMP and MPI. The software architecture has been designed with two major objectives: being easy to maintain and easy to understand. The code is extremely documented and the naming convention fully respected. Driven by its user-oriented philosophy, ScalFMM is using CMAKE as a compiler/installer tool. Even if ScalFMM is written in C++ it will support a C and fortran API soon.

The ScalFMM package is very much a first outcome of the research activity described in Section 3.4.
5.7. Other software

These software packages are or will be developed in collaboration with some academic partners (LIP6, LaBRI, CPMOH, IPREM, EPFL) or in collaboration with industrial partners (CEA, TOTAL, EDF) in the context of their private R&D or production activities.

- For the materials physics applications, a lot of development will be done in the context of ANR projects (NOSSI and proposal OPTIDIS, see Section 4.2) in collaboration with LaBRI, CPMOH, IPREM, EPFL and with CEA Saclay and Bruyère-le-Châtel.

- In the context of the PhD thesis of Mathieu Chanaud (collaboration with CEA/CESTA), we have developed a new parallel platform based on a combination of a geometric full multigrid solver and a direct solver (the PaStiX solver developed in the previous ScAlApplix project-team) to solve huge linear systems arising from Maxwell equations discretized with first-order Nédelec elements (see Section 3.3).

- Finally, we contribute to software developments for seismic analysis and imaging and for wave propagation in collaboration with TOTAL (use of GPU technology with CUDA).
5. Software

5.1. DiaSuite: a Development Environment for Sense/Compute/Control Applications

**Participants:** Charles Consel [correspondent], Benjamin Bertran, Ghislain Deffrasnes, Amélie Marzin, Damien Cassou, Julien Bruneau, Emilie Balland.

Despite much progress, developing a pervasive computing application remains a challenge because of a lack of conceptual frameworks and supporting tools. This challenge involves coping with heterogeneous devices, overcoming the intricacies of distributed systems technologies, working out an architecture for the application, encoding it in a program, writing specific code to test the application, and finally deploying it.

**DiaSuite** is a suite of tools covering the development life-cycle of a pervasive computing application:

- *Defining an application area.* First, an expert defines a catalog of entities, whether hardware or software, that are specific to a target area. These entities serve as building blocks to develop applications in this area. They are gathered in a taxonomy definition, written in the taxonomy layer of the **Diaspec** language.

- *Designing an application.* Given a taxonomy, the architect can design and structure applications. To do so, the **Diaspec** language provides an application design layer [33]. This layer is dedicated to an architectural pattern commonly used in the pervasive computing domain [24]. Describing the architecture application allows to further model a pervasive computing system, making explicit its functional decomposition.

- *Implementing an application.* We leverage the taxonomy definition and the architecture description to provide dedicated support to both the entity and the application developers. This support takes the form of a Java programming framework, generated by the **DiaGen** compiler. The generated programming framework precisely guides the developer with respect to the taxonomy definition and the architecture description. It consists of high-level operations to discover entities and interact with both entities and application components. In doing so, it abstracts away from the underlying distributed technologies, providing further separation of concerns.

- *Testing an application.* **DiaGen** generates a simulation support to test pervasive computing applications before their actual deployment. An application is simulated in the **Diasim** tool, without requiring any code modification. **Diasim** provides an editor to define simulation scenarios and a 2D-renderer to monitor the simulated application. Furthermore, simulated and actual entities can be mixed. This hybrid simulation enables an application to migrate incrementally to an actual environment.

- *Deploying a system.* Finally, the system administrator deploys the pervasive computing system. To this end, a distributed systems technology is selected. We have developed a back-end that currently targets the following technologies: Web Services, RMI, SIP and OSGI. This targeting is transparent for the application code. The variety of these target technologies demonstrates that our development approach separates concerns into well-defined layers.

This development cycle is summarized in the Figure 1.

See also the web page [http://diasuite.inria.fr](http://diasuite.inria.fr).
5.1.1. DiaSpec: a Domain-Specific Language for Networked Entities

The core of the DIA SUITE development environment is the domain specific language called DIA SPEC and its compiler DIA GEN:

- **DIA SPEC** is composed of two layers:
  - The *Taxonomy Layer* allows the declaration of entities that are relevant to the target application area. An entity consists of sensing capabilities, producing data, and actuating capabilities, providing actions. Accordingly, an entity description declares a data source for each one of its sensing capabilities. As well, an actuating capability corresponds to a set of method declarations. An entity declaration also includes attributes, characterizing properties of entity instances. Entity declarations are organized hierarchically allowing entity classes to inherit attributes, sources and actions. A taxonomy allows separation of concerns in that the expert can focus on the concerns of cataloging area-specific entities. The entity developer is concerned about mapping a taxonomical description into an actual entity, and the application developer concentrates on the application logic.
  
  - The *Architecture Layer* is based on an architectural pattern commonly used in the pervasive computing domain [24]. It consists of context components fueled by sensing entities. These components process gathered data to make them amenable to the application needs. Context data are then passed to controller components that trigger actions on entities. Using an architecture description enables the key components of an application to be identified, allowing their implementation to evolve with the requirements (e.g., varying light management implementations in a controller component to optimize energy consumption).

- **DIA GEN** is the DIA SPEC compiler that performs both static and runtime verifications over DIA SPEC declarations and produces a dedicated programming framework that guides and eases the implementation of components. The generated framework is independent of the underlying distributed technology. As of today, DIA GEN supports multiple targets: Local, RMI, SIP, Web Services and OSGI.

5.1.2. DiaSim: a Parametrized Simulator for Pervasive Computing Applications
Pervasive computing applications involve both software and integration concerns. This situation is problematic for testing pervasive computing applications because it requires acquiring, testing and interfacing a variety of software and hardware entities. This process can rapidly become costly and time-consuming when the target environment involves many entities.

To ease the testing of pervasive applications, we are developing a simulator for pervasive computing applications: DIA SIM. To cope with widely heterogeneous entities, DIA SIM is parameterized with respect to a DIA SPEC specification describing a target pervasive computing environment. This description is used to generate with DIA GEN both a programming framework to develop the simulation logic and an emulation layer to execute applications. Furthermore, a simulation renderer is coupled to DIA SIM to allow a simulated pervasive system to be visually monitored and debugged. The simulation renderer is illustrated in Figure 2.

5.2. DiaSuiteBox: an Open Service Platform

Participants: Benjamin Bertran [correspondent], Julien Bruneau, Charles Consel, Emilie Balland.

The DiaSuiteBox platform runs an open-ended set of applications leveraging a range of appliances and web services. Our solution consists of a dedicated development environment, a certifying application store, and a lightweight runtime platform. This solution is based on the DIA SUITE project.

The DiaSuiteBox platform can be embedded in a small plug-computer. This box can be easily deployed, runs silently, and has a reduced energy consumption. Thanks to the application store and the developer community, the platform is fed by a full offer of new innovative applications. During the submission process, an application is automatically analyzed and checked in order to be certified. The user is ensured of the behavior of its applications are innocuous and correct beside the provided information. This box relies on several technology standards like UPnP, Bluetooth, USB, etc. As shown in Figure 3, this platform can be easily extended by plugging appliances directly on the box or by connecting devices on the local network.

See also the web page http://diabox.inria.fr.

5.3. Pantagruel: a Visual Domain-Specific Language for Ubiquitous Computing

Participants: Ghislain Deffrasnes [correspondent], Julien Mercadal, Charles Consel.
Figure 3. DiaSuiteBox platform architecture

Figure 4. A screenshot of the Pantagruel graphical editor
Pantagruel aims at easing the description of an orchestration logic between networked entities of a pervasive environment. First, the developer defines a taxonomy of entities that compose the environment. This step provides an abstraction of the entities capabilities and functionalities. Second, the developer defines the orchestration logic in terms of rules. To facilitate its programming, we provide a visual domain-specific language based on the sensor-controller-actuator paradigm. An example of a visual orchestration is given in Figure 4 where a shower automatically runs at the right temperature when someone enters the bathroom and closes the door.

Pantagruel brings a high-level layer intended to complement existing tools in the activity of safe orchestration logic description, allowing novice-programmers to prototype pervasive applications. The Pantragruel compiler generates code compliant with the D1ASUITE toolset. Pantagruel is being completed by tools aimed at verifying safety properties like termination and reachability.

See also the web page [http://phoenix.inria.fr/software/pantagruel](http://phoenix.inria.fr/software/pantagruel).
RUNTIME Project-Team

5. Software

5.1. Common Communication Interface

Participant: Brice Goglin.

- The Common Communication Interface aims at offering a generic and portable programming interface for a wide range of networking technologies (Ethernet, InfiniBand, ...) and application needs (MPI, storage, low latency UDP, ...).
- CCI is developed in collaboration with the Oak Ridge National Laboratory and several other academics and industrial partners.
- CCI is in early development and currently composed of 19 000 lines of C.
- http://www.cci-forum.org

5.2. Hardware Locality

Participants: Brice Goglin, Samuel Thibault.

- Hardware Locality (HWLOC) is a library and set of tools aiming at discovering and exposing the topology of machines, including processors, cores, threads, shared caches, NUMA memory nodes and I/O devices.
- It builds a widely-portable abstraction of these resources and exposes it to the application so as to help them adapt their behavior to the hardware characteristics.
- HWLOC targets many types of high-performance computing applications [6], from thread scheduling to placement of MPI processes. Most existing MPI implementations, several resource managers and task schedulers already use HWLOC.
- HWLOC is developed in collaboration with the OPEN MPI project. The core development is still mostly performed by Brice GOGLIN and Samuel THIBAULT from the RUNTIME team-project, but many outside contributors are joining the effort, especially from the OPEN MPI and MPICH2 communities.
- HWLOC is composed of 33 000 lines of C.
- http://runtime.bordeaux.inria.fr/hwloc/

5.3. KNem

Participants: Brice Goglin, Stéphanie Moreaud.

- KNEM (Kernel Nemesis) is a Linux kernel module that offers high-performance data transfer between user-space processes.
- KNEM offers a very simple message passing interface that may be used when transferring very large messages within point-to-point or collective MPI operations between processes on the same node.
- Thanks to its kernel-based design, KNEM is able to transfer messages through a single memory copy, much faster than the usual user-space two-copy model.
- KNEM also offers the optional ability to offload memory copies on INTEL I/O AT hardware which improves throughput and reduces CPU consumption and cache pollution.
- KNEM is developed in collaboration with the MPICH2 team at the Argonne National Laboratory and the OPEN MPI project. These partners already released KNEM support as part of their MPI implementations.
- KNEM is composed of 7000 lines of C. Its main contributor is Brice GOGLIN.
- http://runtime.bordeaux.inria.fr/knem/
5.4. Marcel

Participants: Olivier Aumage, Yannick Martin, Samuel Thibault.

- MARCEL is the two-level thread scheduler (also called N:M scheduler) of the PM2 software suite.
- The architecture of MARCEL was carefully designed to support a large number of threads and to efficiently exploit hierarchical architectures (e.g. multicore chips, NUMA machines).
- MARCEL provides a seed construct which can be seen as a precursor of thread. It is only when the time comes to actually run the seed that MARCEL attempts to reuse the resources and the context of another, dying thread, significantly saving management costs.
- In addition to a set of original extensions, MARCEL provides a POSIX-compliant interface which thus permits to take advantage of it by just recompiling unmodified applications or parallel programming environments (API compatibility), or even by running already-compiled binaries with the Linux NPTL ABI compatibility layer.
- For debugging purpose, a trace of the scheduling events can be recorded and used after execution for generating an animated movie showing a replay of the execution.
- The MARCEL thread scheduling library is made of 80 000 lines of code.
- http://runtime.bordeaux.inria.fr/marcel/
- Marcel has been supported for 2 years (2009-2011) by the INRIA ADT Visimar.

5.5. ForestGOMP

Participants: Olivier Aumage, Yannick Martin, Pierre-André Wacrenier.

- FORESTGOMP is an OPENMP environment based on both the GNU OPENMP run-time and the MARCEL thread library.
- It is designed to schedule efficiently nested sets of threads (derived from nested parallel regions) over hierarchical architectures so as to minimize cache misses and NUMA penalties.
- The FORESTGOMP runtime generates nested MARCEL bubbles each time an OPENMP parallel region is encountered, thereby grouping threads sharing common data.
- Topology-aware scheduling policies implemented by BUBBLESCHED can then be used to dynamically map bubbles onto the various levels of the underlying hierarchical architecture.
- FORESTGOMP allowed us to validate the BUBBLESCHED approach with highly irregular, fine grain, divide-and-conquer parallel applications.
- http://runtime.bordeaux.inria.fr/forestgomp/

5.6. Open-MX

Participants: Brice Goglin, Ludovic Stordeur.

- The OPEN-MX software stack is a high-performance message passing implementation for any generic ETHERNET interface.
- It was developed within our collaboration with Myricom, Inc. as a part of the move towards the convergence between high-speed interconnects and generic networks.
- OPEN-MX exposes the raw ETHERNET performance at the application level through a pure message passing protocol.
- While the goal is similar to the old GAMMA stack [58] or the recent iWarp [57] implementations, OPEN-MX relies on generic hardware and drivers and has been designed for message passing.
• OPEN-MX is also wire-compatible with Myricom MX protocol and interface so that any application built for MX may run on any machine without Myricom hardware and talk other nodes running with or without the native MX stack.

• OPEN-MX is also an interesting framework for studying next-generation hardware features that could help ETHERNET hardware become legacy in the context of high-performance computing. Some innovative message-passing-aware stateless abilities, such as multiqueue binding and interrupt coalescing, were designed and evaluated thanks to OPEN-MX [23], [10].

• Brice GOGLIN is the main contributor to OPEN-MX. The software is already composed of more than 45,000 lines of code in the Linux kernel and in user-space.

• http://open-mx.org/

5.7. StarPU

Participants: Cédric Augonnet, Nicolas Collin, Nathalie Furmento, Cyril Roelandt, Samuel Thibault, Ludovic Courtès.

• STARPU permits high performance libraries or compiler environments to exploit heterogeneous multicore machines possibly equipped with GPGPUs or Cell processors.

• STARPU offers a unified offloadable task abstraction named codelet. In case a codelet may run on heterogeneous architectures, it is possible to specify one function for each architectures (e.g. one function for CUDA and one function for CPUs).

• STARPU takes care to schedule and execute those codelets as efficiently as possible over the entire machine. A high-level data management library enforces memory coherency over the machine: before a codelet starts (e.g. on an accelerator), all its data are transparently made available on the compute resource.

• STARPU obtains portable performances by efficiently (and easily) using all computing resources at the same time.

• STARPU also takes advantage of the heterogeneous nature of a machine, for instance by using scheduling strategies based on auto-tuned performance models.

• STARPU can also leverage existing parallel implementations, by supporting parallel tasks, which can be run concurrently over the machine.

• STARPU provides a reduction mode, which permit to further optimize data management when results have to be reduced.

• STARPU provides integration in MPI clusters through a lightweight DSM over MPI.

• STARPU comes with a plug-in for the GNU Compiler Collection (GCC), which extends languages of the C family with syntactic devices to describe STARPU’s main programming concepts in a concise, high-level way.

• http://runtime.bordeaux.inria.fr/StarPU/

5.8. NewMadeleine

Participants: Alexandre Denis, François Trahay, Raymond Namyst.

• NEWMADELEINE is communication library for high performance networks, based on a modular architecture using software components.

• The NEWMADELEINE optimizing scheduler aims at enabling the use of a much wider range of communication flow optimization techniques such as packet reordering or cross-flow packet aggregation.
NEWMADELEINE targets applications with irregular, multifold communication schemes such as found in the increasingly common application conglomerates made of multiple programming environments and coupled pieces of code, for instance.

- It is designed to be programmable through the concepts of optimization strategies, allowing experimentations with multiple approaches or on multiple issues with regard to processing communication flows, based on basic communication flows operations such as packet merging or reordering.

- The reference software development branch of the NEWMADELEINE software consists in 90 000 lines of code. NEWMADELEINE is available on various networking technologies: Myrinet, Infini-band, Quadrics and ETHERNET. It is developed and maintained by Alexandre Denis.

- http://runtime.bordeaux.inria.fr/newmadeleine/

5.9. PadicoTM

Participants: Alexandre Denis.

- PadicoTM is a high-performance communication framework for grids. It is designed to enable various middleware systems (such as CORBA, MPI, SOAP, JVM, DSM, etc.) to utilize the networking technologies found on grids.

- PadicoTM aims at decoupling middleware systems from the various networking resources to reach transparent portability and flexibility.

- PadicoTM architecture is based on software components. Puk (the PadicoTM micro-kernel) implements a light-weight high-performance component model that is used to build communication stacks.

- PadicoTM component model is now used in NEWMADELEINE. It is the cornerstone for networking integration in the projects “LEGO” and “COOP” from the ANR.

- PadicoTM is composed of roughly 60 000 lines of C.

- PadicoTM is registered at the APP under number IDDN.FR.001.260013.000.S.P.2002.000.10000.

- http://runtime.bordeaux.inria.fr/PadicoTM/

5.10. MAQAO

Participants: Denis Barthou, Andres Charif-Rubial.

- MAQAO is a performance tuning tool for OpenMP parallel applications. It relies on the static analysis of binary codes and the collection of dynamic information (such as memory traces). It provides hints to the user about performance bottlenecks and possible workarounds.

- MAQAO relies on binary codes and inserts probes for instrumentation directly inside the binary. There is no need to recompile. The static/dynamic approach of MAQAO analysis is the main originality of the tool, combining performance model with values collected through instrumentation.

- MAQAO has a static performance model for x86 architecture and Itanium. This model analyzes performance of the predecoder, of the decoder and of the different pipelines of the x86 architecture, in particular for SSE instructions.

- The dynamic collection of data in MAQAO enables the analysis of thread interactions, such as false sharing, amount of data reuse, runtime scheduling policy, ...

- MAQAO is in the project “ProHMPT” from the ANR. A demo of MAQAO has been made in Jan. 2010 for SME/INRIA days and in Nov. 2010 at SuperComputing, INRIA Booth.

- http://www.maqao.org/
5.11. QIRAL

**Participant:** Denis Barthou.

- QIRAL is a high level language (expressed through LaTeX) that is used to described Lattice QCD problems. It describes matrix formulations, domain specific properties on preconditionings, and algorithms.
- The compiler chain for QIRAL can combine algorithms and preconditionings, checking validity of the composition automatically. It generates OpenMP parallel code, using libraries, such as BLAS.
- This code is developed in collaboration with other teams participating to the ANR PetaQCD project.

5.12. TreeMatch

**Participants:** Emmanuel Jeannot, Guillaume Mercier.

- **TREEMATCH** is a library for performing process placement based on the topology of the machine and the communication pattern of the application.
- **TREEMATCH** provides a permutation of the processes to the processors/cores in order to minimize the communication cost of the application.
- Important features are: the number of processors can be higher than the number of processes; it assumes that the topology is a tree and does not require valuation of the topology (e.g. communication speed); it implements different placement algorithms that are switched according to the input size.
- **TREEMATCH** is implemented as a load-balancer in Charm++ and as an tool for performing rank reordering in OpenMPI and MPICH-2 [37].
5. Software

5.1. Perception Tools

Participants: David Filliat [correspondant], Natalia Lyubova.

5.1.1. Perception Abstraction Engine

Participants: David Filliat [correspondant], Natalia Lyubova.

PAE (Perception Abstraction Engine) is a C++ library developed to provide a uniform interface to existing visual feature detectors such as SIFT, SURF, MSER, superpixels, etc... Its main goal is to be able to use these various feature detectors in a "bag of feature" approach for applications such as robot localisation and object recognition. Several approach are also implemented for the visual vocabularies, in particular the fast incremental vocabularies developed in the team.

The library provide common C++ interfaces to feature detectors, visual features and visual vocabularies. A factory approach make it possible to change the feature detectors and visual vocabularies types and parameters through configuration strings, without the need to recompile. Some applications are also included in the library, in particular topological robot localization (room recognition) and visual object recognition. An Urbi interface is also provided for these modules.

5.1.2. Incremental object discovery

Participants: Natalia Lyubova [correspondant], David Filliat.

This software makes it possible to detect, model and recognize objects in a scenario of interaction between a humanoid robot and a human teacher. It is based either on standard images, or on the kinect camera to take advantage of the depth information. The software is written in C++ and relies mainly on PAE and OpenCV.

The software implements several modules: candidate object segmentation based on motion information, keypoint-based object tracking, incremental object model construction integrating multiple features ( keypoints + superpixels) and object categorisation based on mutual information with robot motors (making it possible to segment robot parts, objects and humans).

![Figure 1. System Overview of the Incremental object discovery Software.](image)

5.2. Learning Algorithms

5.2.1. Neural online learning library

Participant: Alexander GEPPERTH [correspondant].
nnLib is a C/Python-based library for the efficient simulation of neural online learning algorithms. The core user API is implemented in Python as an object-oriented hierarchy, allowing the creation of neural network layers from configuration files in a completely opaque way, as well as the adaptation of multiple parameters at runtime. Available learning algorithms are: PCA (subspace rule and stochastic gradient ascent), sparse coding, self-organizing map, logistic regression and several variants of Hebbian learning (normalized, decaying, ...). nnLib is under development and will be made available to the public under the GPL in 2012.

5.2.2. RLPark - Reinforcement Learning Algorithms in JAVA

Participant: Thomas Degris [correspondant].

RLPark is a reinforcement learning framework in Java. RLPark includes learning algorithms, state representations, reinforcement learning architectures, standard benchmark problems, communication interfaces for three robots, a framework for running experiments on clusters, and real-time visualization using Zephyr. More precisely, RLPark includes:

- Online Learning Algorithms: Sarsa, Expected Sarsa, Q-Learning, Actor-Critic with normal distribution (continuous actions) and Boltzmann distribution (discrete action), average reward actor-critic, TD, TD(λ), GTD(λ), GQ(λ), TDC
- State Representations: tile coding (with no hashing, hashing and hashing with murmur2), Linear Threshold Unit, observation history, feature normalization, radial basis functions
- Interface with Robots: the Critterbot, iRobot Create, Nao
- Benchmark Problems: mountain car, swing-up pendulum, random walk, continuous grid world

An example of RLPark running an online learning experiment on a reinforcement learning benchmark problem is shown in Figure 2.

RLPark was started in spring 2009 in the RLAI group at the university of Alberta (Canada) when Thomas Degris was a postdoc in this group. RLPark is still actively used by RLAI. Collaborators and users include Adam White (patches for bug fixes, testing), Joseph Modayil (implementation of the NAO interface, patches for bug fixes, testing) and Patrick Pilarski (testing) from the University of Alberta. RLPark has also been used by Richard Sutton, a professor and iCORE chair in the department of computing science at the University of Alberta, for a demo in his invited talk Learning from Data at the Neural Information Processing Systems (NIPS) 2011. Future developments include the implementation of additional algorithms (the Dyna architecture, back propagation in neural networks, ...) as well as optimizations of vector operations using GPU (with OpenCL) and additional demos. Future dissemination includes a paper in preparation for the JMLR Machine Learning Open Source Software. Documentation and tutorials are included on the http://thomasdegris.github.com/rlpark/ RLPark web site. RLPark is licensed under the open source Eclipse Public License.

5.2.3. Autonomous or Guided Explorer (AGE)

Participant: Sao Mai NGUYEN [correspondant].

The "Autonomous or Guided Explorer" program is designed for the systematic evaluation and comparison of different exploration mechanisms allowing a simulated or a real robot to learn and build models by self-exploration or social learning. Its conception allows an easy selection of different intrinsically motivated exploration or classical social learning mechanisms. Are provided algorithms such as Random Exploration, SAGG-RIAC, SGIM-D, imitation learning, learning by Observation. The program uses the new objet-oriented programming capability of Matlab, to enhance flexibility and modularity. The main program is built around objects that represent the different modules and the general architecture of such learning algorithms: action space exploration, goal space exploration, interaction with a human, robot control, model computation, but also evaluation and visualisation modules.
Figure 2. An example of an experiment in RLPark. Zephyr displays two views of a learned weight vector, an animation of the problem, the current policy distribution learned by the algorithm and the reward obtained by the algorithm.

The software is designed to easily tune learning parameters and to be easily plugged to other robotic setups. Its object-oriented structure allows safe adaptation to different robotic setups, learning tasks where the structure of the model to learn differs, but also different action or goal spaces. This program is used by Sao Mai Nguyen of the team to compare the performance of different learning algorithms. These results were partly published in [27]. Future work will take advantage of its flexibility and implement new default robotic setups, robot control, action and goal spaces, and most of all, new types of interaction with a human.

5.2.4. NMF Python implementation

Participant: Olivier Mangin [correspondant].

This library is meant to implement various algorithms for Nonnegative Matrix Factorization in the Python programming language, on top of the Numpy and Scipy scientific libraries.

Some Python NMF libraries already exist, such as the one present in the scikit-learn project. However, most of them are quite limited in comparison to recent advances in these techniques (for example, extension of NMF algorithms to wider families of penalties such as the beta-divergence family). On the other hand, existing MATLAB software has been released by the authors of some of these algorithms but, first, code is not available for every interesting algorithm and none of those various pieces of code implements the whole set of features that one would like to use.

This project is in a very early stage and yet only for internal use in the team. It could, however, be released in the future, for example, integrated in the previously mentioned scikit-learn project.

5.3. Software Platforms

5.3.1. JBox2D wrapper

Participant: Fabien BENUREAU [correspondant].
ProcBox2D is a wrapping of Processing and JBox2D to satisfy common robotic research needs. In order to quickly prototype research ideas, a simple and efficient simulation framework is of great use. JBox2D is a 2D rigid-body physic engine. Written in Java, it is very fast, typically allowing to compute simulation 60 times faster than real time. Mass simulations can be carried in a timely manner, and improving the process of iterating the conception and implementation of new algorithms. Processing is a graphical framework in Java, and is used to display the simulations of JBox2D. An example of a simulation rendering is visible in Figure 3.

![Figure 3. A JBox2D simulation rendered with Processing using ProcBox2D. A robotic arm is interaction with dynamic object (in pink and yellow); the environment contains obstacles and walls (in dark purple).](image)

While several libraries exist that expose the JBox2D engine to the Processing framework, they suffer from binding Processing irrevocably into the experiment description. As such, simulations without a graphical context, a situation commonly encountered on remote servers and computing clusters are impossible using these libraries. ProcBox2D was written to fill this gap. It allows the conception of experiments to be done using Processing display capability, while, later one, without modifications of the code, to execute the simulations without any dependency to Processing, on a cluster for instance. The use of Processing allows interactions with the scene via the mouse, which makes ProcBox2D a potential tool in demonstration or imitation learning experiments.

ProcBox2D also provides a sensor and controller interface. Actuated joints can be controlled in torque and velocity, and a PID controller for position control is planned. ProcBox2D implementation begun in November 2011 and was presented and made available to the team in December 2011. It is expected that it will increase productivity of researchers that previously had to work out a solution for themselves, often using in early stage of research complex and time-consuming simulation frameworks.

### 5.3.2. V-REPBridge

**Participant:** Paul FUDAL [correspondant].

V-REPBridge (formerly uV-REPBridge) is a set of software tools to control V-REP through an external client; it consists of a plugin for V-REP and an API to control V-REP.

V-REP - the Virtual Robot Experimentation Platform - is a robot simulator which allows the editing and simulation of robotic systems and sub-systems. Also, it can be integrated and combined using a complete API.
V-REPBridge is a way to interact with a simulation loaded through an Urbi script or a Python application. Based on network communication, V-REPBridge can be used locally (V-REP and the client on the same computer) or remotely. The V-REP simulator’s main use is to perform experiments with virtual robots and environments. But, because V-REPBridge API provides classic functionality like, for example, setting position of a joint or its torque, getting sensor value, etc... an existing application built on top of V-REPBridge can be easily repurposed to use the interface of a real robots.

The development of the plugin for V-REP is made under Windows environment using the V-REP and Windows API. The plugin acts as a server to which a client can connect in order to control the simulation. The client is provided as an API written in C++. This API is available for Windows, Mac and Linux and bindings are available for UrbiScript and Python. The bindings are based on the Urbi API and the Boost Python Library.

Today, V-REPBridge is fully functional and already used in several research experiments, and provide more than 130 V-REP API functions which can be called by the client; here is a non-exhaustive list of V-REP functionalities available in the client:

- joint functionality (position, velocity, torque, etc...),
- object functionality (position, orientation, etc...),
- force sensor functionality,
- inverse kinematic and geometric functionality,
- proximity sensors functionality,
- collision detection functionality,
- minimum distance calculation functionality,
- path planning functionality,
- dynamic functionality,
- ...

V-REPBridge is also provided with an user documentation which includes some howtos (build, use), a complete list of available functions (with synopsis and parameters/returned value description) and some short examples written in Urbi and Python.

Finally, a developer documentation will be available soon to help developers who wants to implement missing V-REP calls both in the plugin and the client, or wants to implements theirs owns functions callable in the client.

The development of V-REPBridge was started at the beginning of year 2011. First release was made in February for testing and debugging foundation of the software. After this short period, time was spent expanding the software and adding new functionalities to bring a response to the needs of the team.

First experiments with V-REPBridge was made for IJCAI in july (Mai NGUYEN), ICDL in august (Mai NGUYEN/Matthieu LAPEYRE) and Humanoid in october 2011 (Matthieu LAPEYRE). It was a good feedback for improving the performance and to identify potential improvements.

Work is still in progress for minor bugfixes, support of V-REP minor releases and preparation of the future version of V-REP which will run not only Windows but also on Linux and Mac OS X. A first private beta of V-REP 3 will be available at the end of january.

### 5.3.3. Rhoban Move Studio

**Participants:** Olivier Ly [correspondant], Hugo Gimbert, Jérôme Béchu, Paul Fudal.

#### 5.3.3.1. Main software stack

RhobanMoveStudio is a software suite to easily create and control robots, Acroban, FLOWERS Fields/Robot Lamps and Ergorobots/FLOWERS Fields in particular.

This platform has already been presented last year, but it has evolved, in particular for the motor control part. The software architecture has been kept similar but performance has been improved.
Figure 4. The complete software architecture of Rhoban Move Studio
The system runs on an electronic board (based on ARM9 processor) and uses a linux distribution (OpenWrt). The software is composed of several layers:

- **Kernel module** The role of the module is to implement the electronic communication with devices. It enables to manage Dynamixel\(^1\) motors, generates PWM\(^2\) signals, uses digital readers/writers, I2C bus and more. This year the motor communication have been significantly improved and gained support for accelerometers. This module is designed to run in root mode, to guarantee execution without system interruption, as required by robotic application.

- **Low level** This set of functions is used to communicate with the module through a dedicated shared memory.

- **Move Scheduler** This library provides enables a high level specification of low level motor control loop based on graph of input/output blocks (see Section 5.3.3.2).

- **Rhoban server** This software offers access to the full API of rhoban features through a TCP Socket.

- **Librhoban** This TCP client library provides communication with the Rhoban Server and thus to the whole API. It is a dynamic library, thread safe and secure.

Except for the kernel module which is written in C ANSI, this softwares are written in C++.

5.3.3.2. **Move Scheduler**

Recently (October 2011) a new layer was added to the software. Its role is to enable low level motor control loops through a high level representation.

This representation introduces the concept of blocks. Each block is a computing unit with inputs and outputs. The output of a block can be the input of another one, thus forming a graph of interaction between those unit. Each block is a function (for example addition, multiplication, derivation, integration, spline generation). Special blocks are also provided for sensor inputs and motor outputs.

Graphical interface was developed to easily design such movements. It is called *Move Scheduler Modeler*, and written in Python (PyQt). This software has import/export capabilities to XML files.

5.3.4. **UFlow**

**Participant:** Jérome Béchu [correspondant].

We developed some new UObjects to enrich the UFlow Toolbox. The UFlow Toolbox is a collection of various software modules for programming and scripting robot sensorimotor loops, aimed at allowing rapid prototyping in the FLOWERS team, and integrated in the URBI framework. URBI, developed by GOSTAI, supports the integration of heterogeneous robotic software modules. It uses a dynamic scripting language, which manages parallel and event processing. Each module, called UObject, is written in C++. We still continue to develop this collection of UObjects for the team.

5.3.4.1. **USoundManager**

This UObject is used to play sound. It’s possible to update the sound while playing. This new version is already based on FMOD.

A new version has just been made. Based on OpenAL, this UObject has the exact same interface as the previous one expect that we include a media manager. With this functionality we load just one time the same sound (We keep it in memory a dictionary of sounds).

5.3.4.2. **URhoban**

wrap the API of the librhoban (see the previous chapter). This tool is especially develop to control Bioloid motors in high frequency. With that software we can create instance of motors scanned and directly read and write features like position, torque, load, speed.

---

\(^1\)Broadly used servo motor product from Robotis [http://www.robotis.com](http://www.robotis.com)

\(^2\)Electronic signal : Pulse With Modulation
5.3.4.3. **UXml**

is an UObject based on TinyXml. It is designed to quickly save and restore URBI List in a xml file. It is generally used to store/load parameters like the list of motors in the ErgoRobot platform.

5.3.4.4. **USmartLed**

was created to use the LinkM USB Device to control RGB lights. It is based on the linkm driver (modified to support multiple USB devices). We can control intensity of each light for each primary color.

5.3.4.5. **UGui**

is designed to draw basic 2D primitives. A new version based on SFML was developed this year. It is used in the ErgoRobot project to run a simulation of the setup with a graphical interface.

5.3.4.6. **USqlite**

is an UObject to wrap functionalities of SQLite in URBI. SQLite is a software library that implements a tiny SQL database engine.

5.3.4.7. **UNamingGame**

is UObject used to play the Naming Game. The Naming Game is an algorithm based on communication between agents, who progressively agree meanings of words.

5.3.5. **ErgoRobot/Flowers Field Software**

**Participants:** Jérôme Béchu [correspondant], Pierre-Yves Oudeyer, Pierre Rouanet, Olivier Mangin, Fabien Benureau, Mathieu Lapeyre.

In the context of its participation to the exhibition “Mathematics: A Beautiful Elsewhere” at Fondation Cartier pour l’Art Contemporain in Paris, starting from 19th October 2011 and to be held until 18th March 2012, the team has elaborated and experimented a robotic experimental set-up called “Ergo-Robots/FLOWERS Fields”. This set-up is not only a way to share our scientific research on curiosity-driven learning, human-robot interaction and language acquisition with the general public, but, as described in the Results and Highlights section, attacks a very important technological challenge impacting the science of developmental robotics: How to design a robot learning experiment that can run continuously and autonomously for several months?
The global scenario for the robots in the installation/experiment is the following. In a big egg that has just opened, a tribe of young robotic creatures evolves and explores its environment, wreathed by a large zero that symbolizes the origin. Beyond their innate capabilities, they are outfitted with mechanisms that allow them to learn new skills and invent their own language. Endowed with artificial curiosity, they explore objects around them, as well as the effect their vocalizations produce on humans. Human, also curious to see what these creatures can do, react with their own gestures, creating a loop of interaction which progressively self-organizes into a new communication system established between man and ergo-robots.

We now outline the main elements of the software architectures underlying this experimental setup.

5.3.5.1. System components

The software architecture is organized to control the experiment at several levels, and in particular:

- **Scenes**: The organization of behavioural scenes, managing the behaviours that are allowed to each robot at particular times and in particular contexts;
- **Behaviours**: The individual behaviours of robots, also called stems, which are outlined in the next section;
- **stems**: The low-level actions and perception of robots while executing their behaviours, including motors control on the five physical stems, color and intensity of lights inside the stem head, production of sounds through speakers. Sensors are the kinect used to interact with visitors, and motor feedback capabilities.

In addition to that a video projector is used to display some artistic view of stem agents internal state.

![Diagram of Stem, Behaviour, and Scene components](image)

**Figure 6. Three important concepts in ErgoRobots**

5.3.5.2. Behaviours

A number of innate behaviours were designed and are used by the robots as elementary behaviours of more complex behaviours, including the three following learning behaviours.

- **The Naming Game** is a behaviour played by stems two-by-two and based on computational models of how communities of language users can self-organize shared lexicons. In the naming game, stems interact with each other in a stylised interaction. Repeated interactions lead to the development of a common repertoire of words for naming objects. More precisely, object belong to meaning spaces. Two such spaces have been implemented for the exhibition. The first one is related to object spatial categorization and the second one is related to movement categorization. The object space contains stems, some hole in walls and the interaction zone. The movement space contains representations of small dances that stem can produce and reproduce.

- **Object Curiosity** is a behaviour in controlling intrinsically motivated exploration of the physical environment by the stems. A small wood object is present in the reachable physical environment of the stem, attached on the top of a spring so that it is guaranteed that it comes back to its original position. The stem uses a motor primitive to act on the object and motor feedback to detect movements of the object. The robot learns through active exploration what kind of parameters motor primitive will result in touching the object.
Figure 7. A Stem with the head designed by David Lynch and an Object
*Birds Curiosity* is a behaviour that drives robots to explore, through curiosity-driven learning, interaction with humans. One stem, generally the stem in the center, plays a sound, predicts the visitor reaction, look the interaction zone and wait the gesture of the visitor. To produce a sound the visitor have to make a gesture in space. In the next iterations, the robot chooses to produce sounds to human which produce most surprising responses from the human (i.e. the robot is “interested” to explore sound interactions which are not easily predictable by itself). As describe in the picture, the space is split in four. Each zone corresponding with a sound.

![Figure 8. A virtual visitor interact with a virtual grid](image)

5.3.5.3. *Programming tools*

The system is based on URBI and used some UObjects from UFlow. The most important part of the system is written in URBI script. Python and freenect are used too.

The system at the startup detects motors and lights. It create dynamically a list of Stem. A Stem is one robot with 6 motors as described in hardware part.

To interact with people, we used the freenect library to interface with the kinect, with a binding to python where detection and following of gestures is made.

For the display, we display an abstract rendering of the structure inside each ErgoRobot, using a python parser to read and parse log file from the ErgoRobot system, and the Bloom/Processing software to create and display the rendering. Currently, the system has three displays, one for the naming game, another one for birds curiosity and the last one for objects curiosity.

The sound system used the UObject USoundManager. It plays sounds when required by a behaviour, it also plays word sounds in Naming Game behaviour.

The Light system used Linkm technologies. In the head of each ErgoRobot we put two lights devices. Each light device is a RGB Light. We can control the intensity of each primary color through I2C control. To control lights we used LinkM USB Device. And finally we used an UObject dedicated to communicate with the USB Device.

3 Kinect library
5.3.5.4. Maintenance

A dedicate maintenance software is used to switch off, switch on the system. This software is written in Python (and Qt). The status of ErgoRobots is display on the graphical interface. Buttons are present too: Start, Stop, Reset and Take a video.

Recently we added a video system to have a visual feedback of motors usage and also to detect eventual problems. This is a screenshot of the application:

5.4. Visualization Tools

5.4.1. Zephyr - Visualization Platform

Participant: Thomas Degris [correspondant].

Zephyr is a software in Java and Eclipse Rich Client Platform to visualize numeric variables and data structure in real time and at different time scale. Zephyr is practical for developers because it requires only minimal changes in the code to debug: it uses Java reflexivity to automatically detect variables in the code to monitor and data structure with an associated dedicated view. Zephyr can easily be extended with new plugins because it is based on the popular Eclipse Rich Client Platform. Consequently, Zephyr takes advantage of already existing and fully operational Eclipse plugins for many of its functionalities. Finally, Zephyr is distributed with a Java python virtual machine named Jython and a lisp implementation named Clojure. An example of a Zephyr screen is shown in Figure 11.

Zephyr was started in fall 2009 in the RLAI group at the university of Alberta (Canada) when Thomas Degris was a postdoc in this group. Zephyr is still actively used by RLAI. Users include Adam White, Joseph Modayil and Patrick Pilarski from the University of Alberta. Moreover, Zephyr has been registered on the Eclipse marketplace since October 2011 where it has been downloaded a few times by anonymous users. Future dissemination includes the implementation of demos and tutorial videos. Documentation about Zephyr is included on http://thomasdegris.github.com/zephyr/. Zephyr is licensed under the open source Eclipse Public License.

5.4.2. Bloom - particle-based physical engine

Participants: Fabien BENUREAU [correspondant], Olivier Mangin [correspondant].
Figure 10. Maintenance Software for the ErgoRobots.

Figure 11. An example of Zephyr showing the different steps of a video processing pipeline in real-time.
Bloom is a particle-based physical engine that was coded for the Ergorobot exhibition. Written in a matter of days in September 2011, Bloom is based on Processing and coded in Java. It is currently running all of the projected visualisation of the Ergorobot installation. Bloom greatest strength is to provide an intuitive and lightweight tool to display of complex and dynamic information, such as the morphology of a robot vocabulary, as shown in Figure 12. As such, it should permit to examine the state of complex data structure in real-time during experiment, getting insights and allow detection and tracking of issues in algorithms being developed. Bloom is a great complement in research work to the use of charts and graphs. Bloom has since been made available and presented to the team in December 2011.

Figure 12. The vocabulary of 2 interacting robots of the ErgoRobot installation. The blue particles represent meanings, while the orange ones represent words. The strength of association between them is represented by the length of the edges linking them (shorter is stronger). It easy to spot the presence of synonyms and of difference of topology in the vocabulary of the two robots.

5.5. Hardware

5.5.1. The Ergo-Robots Hardware Platform

Participants: Jerome BECHU [correspondent], Fabien BENUREAU, Haylee FOGG, Hugo GIMBERT, Matthieu LAPEYRE, Olivier LY, Olivier MANGIN, Pierre-Yves OUDEYER, Pierre ROUANET.

ErgoRobots is a hardware platform for showcasing a number of curiosity and learning behaviours for the public to interact with. The platform can also have future uses inside the lab for experiments that require more than one robot to complete. Although this system is entirely new this year, a very different previous version existed with the name FLOWERSField. It consists of five ErgoRobots, a control system, an interaction system, a display system, a sound system and a light system. There is an external system which monitors the ErgoRobots which contains a control system, a power system, a surveillance system and a metric capture system. This system went live on October 19 2011 without lights which will be added in late December.

The Ergo-Robot system: The robots themselves are each composed of six motors (see figure). Currently, the heads of the robots have been created in wax by David Lynch and the entire system is displayed at Fondation Cartier inside a large egg shaped orb as shown in the following diagram. The control system module contains both an MMNET1002 control board with an UART-RS485 breakout board which communicates with a ubuntu Linux PC via an ethernet cable. The mment board communicates with the motors, but all other ErgoRobot systems communicate with the PC directly. The sound system is currently externally provided and communicates with the PC. The light system is a series of two or three BlinkM RGB leds placed...
inside each ErgoRobot head that are controlled through two LinkM USB devices directly with the computer. A Kinect placed in front of the system operates as the means for the public to interact with the platform and communicates directly through USB to the PC. The display system is currently an externally provided projector that projects visualisations of the field’s current state behind the ErgoRobots.

**The external system:** This system allows anyone that is monitoring the system to externally control the ErgoRobots system. The PC with which the software control takes place is a Ubuntu Linux system which communicates with the ErgoRobot control system via an ethernet cable. The ErgoRobot hardware system can be managed by an external power system which includes a 15.5V bench top power supply for the ErgoRobot motors, an external 12V plug in adapter for the mment board, an external 5V plug in adapter for the LED lights which are all controlled via an emergency stop button. The maintenance system can be located out of direct view of the ErgoRobot field as it has a surveillance system: a Kinect that can display the current state of the field. More surveillance is conducted through a metric capture system that communicates with the ErgoRobots to obtain various state values of the ErgoRobots through the motor sensors and other data. This surveillance is not entirely in place as of 2011 and will be implemented in early 2012.

---

**5.5.2. Flowers Field/Robot Lamps**

**Participants:** Jérôme BÉCHU [correspondant], Pierre-Yves OUDEYER, Olivier LY, Fabien BENUREAU.

We continued to develop the FLOWERS FIELDS/Robot Lamps experimental set-up, see Figure 14. This set-up explores new forms and new functions of robotics. When we think of robots, we traditionally have in mind either humanoid robots that look like humans and are supposed to do similar things as humans, or industrial robotic arms which should work in factories. On the contrary, the future may come with unforeseen kinds of robots that may enter our everyday homes: for examples, as houses become themselves intelligent...
with domotics, we could imagine that furnitures themselves could become robots. Chairs, tables, televisions, or lamps may become robots. In FLOWERS FIELDS/Robot Lamps, we show robotic lamps which move like living entities, with their own moods and their own system of interaction. They can be thought to be in houses partly as aesthetic objects, and partly for their social presence. Indeed, not only their movements and sounds are life-like, but they are sensible to human presence and can become interested in looking and interacting with people through those movements and sounds. In the future, we could imagine additionally that these robot lamps could serve as a friendly interface with the numeric world: for example, some gestures may be used towards the lamps to tell their hifi system to play a given song in your library.

This year, a major update of the platform consisted in shifting the whole servomotor technology to the RX Robotis Series, allowing much more robustness and sophistication of control. The software was adapted to these new motors, requiring indeed a new mode of control together with a new electronic board. This installation was demonstrated in march 2011 at the INNOROBO International Summit on Personal Robotics in Lyon.

![Figure 14. Flowers Fields/Robot Lamps](image)

### 5.5.3. Humanoid Robot Torso

**Participant:** Haylee FOGG [correspondant].

The Humanoid Robot Torso is a hardware platform that is intended for use in the lab for either experiments or demonstrations. It consists of a humanoid robot that contains just a torso, arms and head. It is entirely new this year, but it has been updated once during the year. The previous version was inspired by Acrobot and consisted of 20 degrees of freedom. The update began in November of 2011 where 3 degrees of freedom was removed from the spine and one degree of freedom was removed from the head.

The Torso has two arms. At the time of writing one arm consists of a three fingered claw that is controlled by a single motor, and the other is just a flat push mechanism. The arm with the claw contains seven degrees of freedom (including "grip") and the other only five. The torso itself has two degrees of freedom. The head is soon to consist of an iPhone for the face and a separate usb camera for the 'eyes' with the ability to move in two degrees (pitch and roll) in early 2012.

The hardware is bothrobotis Dynamixel RX-28 and R-64 motors attached together with standard robotis frames and a substance called polymorph. Polymorph is used to attach a series of springs and elastic to many
of the degrees of freedom to increase smoothing and absorb backlash of the motors. Polymorph was added in November 2011 to replace the previous versions metal that was tooled in the lab.

A method for controlling the motors of the Torso will be under review in 2012.
5. Software

5.1. Eigen

Participant: Gaël Guennebaud [correspondant].

Keywords:
Web: http://eigen.tuxfamily.org/

Eigen is a fast, versatile, and elegant C++ template library for linear algebra and related algorithms. In particular it provides fixed and dynamic size matrices and vectors, sparse matrices and vectors, matrix decompositions (LU, LLT, LDLT, QR, eigenvalues, etc.), some basic geometry features (transformations, quaternions, axis-angles, Euler angles, hyperplanes, lines, etc.), automatic differentiations, etc. Thanks to expression templates, Eigen provides a very powerful and easy to use API. Explicit vectorization is performed for the SSE (2 and later), Altivec and ARM NEON instruction sets, with graceful fallback to non-vectorized code. Expression templates allow to perform these optimizations globally for whole expressions, and to remove unnecessary temporary objects.

Eigen is already a famous library with about 15000 unique visitors of the website per month, while the mailing list holds about 250 members with a very high traffic (400 message per month). After two years of development since the 2.0 release, we released this year the new major 3.0 version.

- Version: 3.0.4
- Programming language: C++

5.2. Expressive Rendering shaders

Participants: Pascal Barla, Benoit Bossavit.

Shaders developed in the course of our research on expressive rendering have been published under the CeCILL-B license, and distributed on the Animaré project webpage (https://iparla.inria.fr/collaborations/animare/). The goal of such a publication is to let members of the scientific community test and compare with our techniques. This also includes plugins for MeshLab and Nuke.

5.3. Navidget - Easy 3D Camera Positioning from 2D Inputs

Participant: Martin Hachet [correspondant].

Keywords:
Web: https://iparla.inria.fr/software/navidget/

Navidget is a new interaction technique for camera positioning in 3D environments. Unlike the existing POI techniques, Navidget does not attempt to automatically estimate where and how the user wants to move. Instead, it provides good feedback and control for fast and easy interactive camera positioning. Navidget can also be useful for distant inspection when used with a preview window.

This new 3D User interface is totally based on 2D input. As a result, it is appropriate for a wide variety of visualization systems, from small handheld devices to large interactive displays. A user study on TabletPC shows that the usability of Navidget is very good for both expert and novice users. Apart from these tasks, the Navidget approach can be useful for further purposes such as collaborative work and animation.

We have developed a C++/OpenGL library, called LibNavidget, which allows you to integrate Navidget in your own applications. A sample application is included in the package.
5.4. ArcheoTUI

Participants: Patrick Reuter [correspondant], Nicolas Mellado.

Keywords:
ArcheoTUI is a software for the virtual reassembly of fractured archaeological objects via tangible interaction with foot pedal declutching. ArcheoTUI is designed to easily change assembly hypotheses, beyond classical undo/redo, by using a scene graph. The software connects to the database of the broken fragments that are organized in an SQL database. In 2011, we extended the ArcheoTUI software in order to account for a physically-based deformation prototype. Moreover, we integrated multi-touch input with a constraint-based reassembly method.